Writeup

November 27, 2017

1 Traffic Sign Recognition

1.1 Writeup

1.2 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

1.3 Rubric Points

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

1.3.2 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

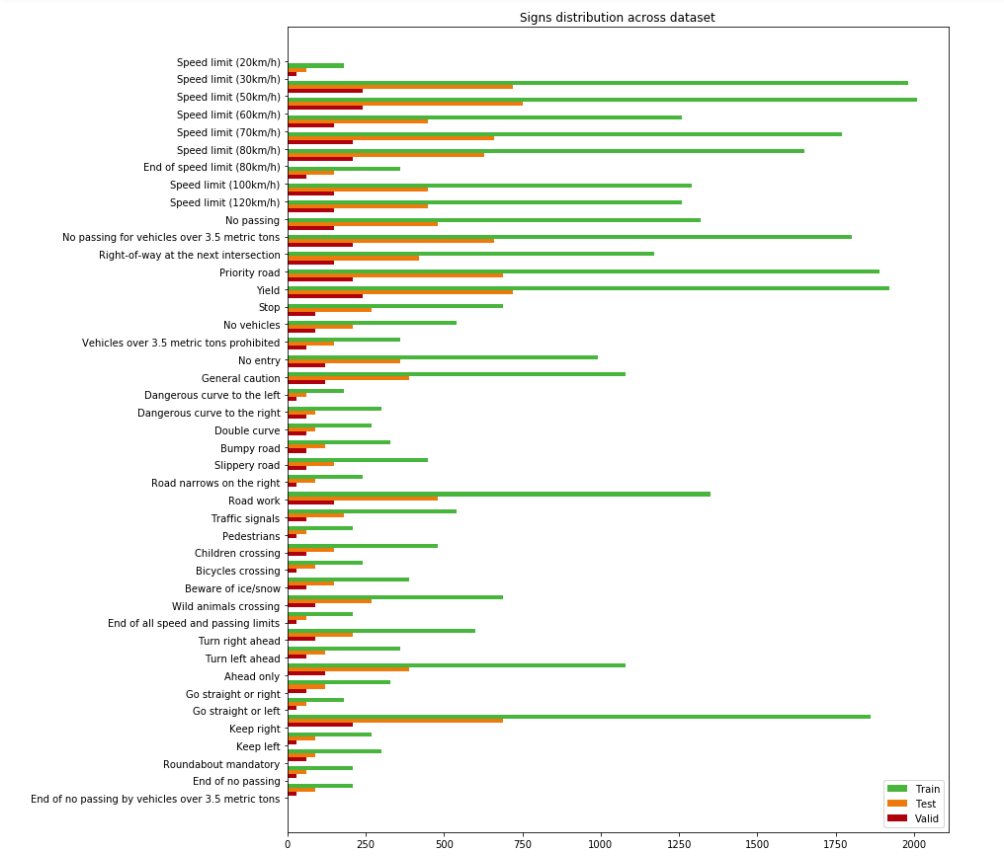
You're reading it! and here is a link to my project code

1.3.3 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

In order to extract some basic info about traffic signs dataset I've used numpy, because it allows to get info about sizes (shapes) of subsets and wraps Python lists with nd.array which allows to perform different kinds of operations on data:

- The size of training set is 34799 images
- The size of the validation set is 4410 images
- The size of test set is 12630 images
- The shape of a traffic sign image is (32, 32, 3)



Signs distribution

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

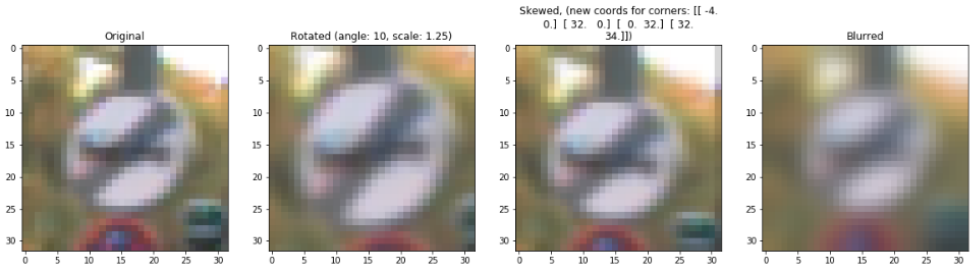
2. Include an exploratory visualization of the dataset.

Here is an exploratory visualization of the data set. It is a bar chart showing how the data is distributed (in terms of classes). This visualization shows how many expamples there are in different subsets (train/valid/test). We can see here that training dataset approximately three times as larger than test dataset and ten times larger than validation dataset. This numbers mathces with the ones mentioned earlier. Also we can see here that some signs ("Speed limit 30" or "Keep right") are significantly more "popular" than the ones like "Road narrows on the right" or "Speed limit 20"

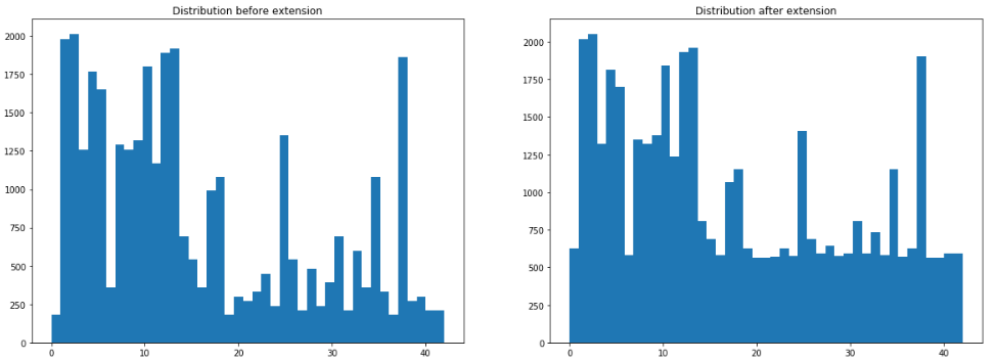
1.3.4 Design and Test a Model Architecture

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.)

For augmenting dataset I've used next techniques of image processing:



Signs augmenting



Distribution comparison

- **image rotation combined with scaling**. Angle is chosen randomly within predefined range. Scaling is used in order to avoid 'blind areas' of black pixels when we perform rotation. Image rotations was chosen because it is allows us to avoid network's overlearning on sign orientation criteria.
- **modifying the perspective** of the image. For each image we randomly shift the corners so that result image skew its perspective. This techinque is used in order to prevent case when network relies on the correct form (circle, triangle and so on) of sign. Images can be skewed because of light refraction, not well-mounted sign, or simply because of position at which an image was taken
- **blurring** image using GaussianBlur. This method was used because we can not rely on the assumption that image was not blurred because of speed or low camera resolution
- combination of blurring and rotation
- combination of blurring and modifying perspective

Note: combination of rotation and perspective modifications is not used, because both of these techniques leads to image scaling. So combinig them we potentially could face a case when image is "overscaled" and only part of sign is visible within initial frame

Techniques described above were used to generate additional training data. For the initial training dataset we perform augmentation "per sign", i.e. the more images available for current sign the less fake images we would create and vice versa. All the generated images as well as labels for them are concatenated with initial dataset. This approach allows to avoid overlearning network on the one type of sign and ignore the other ones.

After extension were made distribution of the training dataset looks like this:

After extending initial dataset with augmented images all the images were converted to the format required for neural network. Two techiques are used for that purpose:

- **grayscaling** is used in order to minimize the noise caused by different light conditions, ISO settings of camera and so on
- **normalization of pixel values**, i.e. converting values so that they belong range from -1 to 1. This conversion is required in order to make the network treating all the features in the same unbiased manner

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 Grayscaled image
Convolution 5x5	1x1 stride, valid padding, outputs 28x28x16
RELU	
Max pooling 2x2	2x2 stride, outputs 14x14x16
Convolution 5x5	1x1 stride, valid padding, outputs 10x10x32
RELU	
Max pooling 2x2	2x2 stride, outputs 5x5x32
Dropout (0.75)	
Flattening	5x5x32 -> 800
Fully connected	Input: 800, Output: 300
RELU	
Fully connected	Input: 300, Output: 70
SIGMOID	
Fully connected	Input: 70, Output: 43
Softmax	

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

For the resulting model following parametrs were used:

- optimizer - AdamOptimizer (it works better than GradientDescent optimizer and also it proved its efficiency in LeNet, the network which solves almost similar problem)
- batch size - 128
- epochs - 25 (the number have been increased in order to allow network to reach higher accuracy)
- learning rate - exponentially decaying with initial value 0.001 and multiplier 0.8 per 800 iterations. It allows to avoid smoothify the process of learning, especially when we are close to the target 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:

- validation set accuracy of 95.4%
- test set accuracy of 93.4%

In order to reach such prediction result an iterative approach was chosen

What was the first architecture that was tried and why was it chosen?

LeNet model was chosen as a starting point. It was chosen because it's been used for the same type of problem (image recognition and classification) and it performs well for this kind of task. Also LeNet model has a very straightforward and clear architecture which allows to play with parameters easily

What were some problems with the initial architecture?

The main problem is the low prediction result for traffic signs. I guess, it is because traffic signs images are much more complicated than the examples of MINIST dataset. In order to correctly detect traffic sign the network should detect colors (or different shades of gray for grayscaled images), shapes of sign, combination of shapes within the sign, the amount of space colored within sign and so on. And LeNet is designed to classify the images where there are only several lines per image that are crucial.

How was the architecture adjusted and why was it adjusted? Typical adjustments could include choosing a different model architecture, adding or taking away layers (pooling, dropout, convolution, etc), using an activation function or changing the activation function. One common justification for adjusting an architecture would be due to overfitting or underfitting. A high accuracy on the training set but low accuracy on the validation set indicates over fitting; a low accuracy on both sets indicates under fitting

First of all I've decided to try to tune the parameters of the initial network. First of all I've increased the number of features extracted on the first layer, because as it is said in the paper the first layer is responsible for the most primitive features and I've thought that if I increase it I'll get more combinations on the next layers which could lead to more accurate predictions. So after the first layer the second one gets it's increased feature number and so on. After I'd finished experimenting with it I've got the accuracy about 90%. The next step was adding a dropout. Dropout after the first layer seemed to be ineffective, but if it is placed after the second one it increases the result of prediction. The next approach I've tried is to change the activation function. Changing functions for convolutions layers decreased the performance, while setting sigmoid function in the very end of the network allowed to reach the target accuracy.

Which parameters were tuned? How were they adjusted and why?

Batch size were tuned. 64, 128, 256 values were applied. It didn't change the performance a lot, but for 128 it gets a hitghest scores.

Learning rate was tuned. As it's been written above, decaying learning rate helped to avoid leaps in performance of the network

Epochs number was increased to 25. After this number accuracy of validation dataset decreases, so, I guess, it is because overfitting to the training dataset. After experimenting I've found that 25 is the optimal value for this network

What are some of the important design choices and why were they chosen? For example, why might a convolution layer work well with this problem? How might a dropout layer help with creating a successful model?

Convolution layers were chosen because they are great at working with images. They allows to extract some features like lines, color gradients, splashes and so on and combinate them later to recognize sophisticated pattern. Dropouts are helpful, 'cause not all found features are crucial, moreover some of them could lead a network to the fraudlent predictions. Dropouts allows to constantly change the set of reliable features and network do not stuck at the local minimum

If a well known architecture was chosen:

What architecture was chosen?

LeNet

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

Because it solves a very similar problem

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

Predictions with the high accuracy are the best evidences that selected model works fine

1.3.5 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:

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
Speed limit (60km/h)	Speed limit (60km/h)
Yield	Yield
Go straight or right	Go straight or right
No entry	No entry
Priority road	Priority road

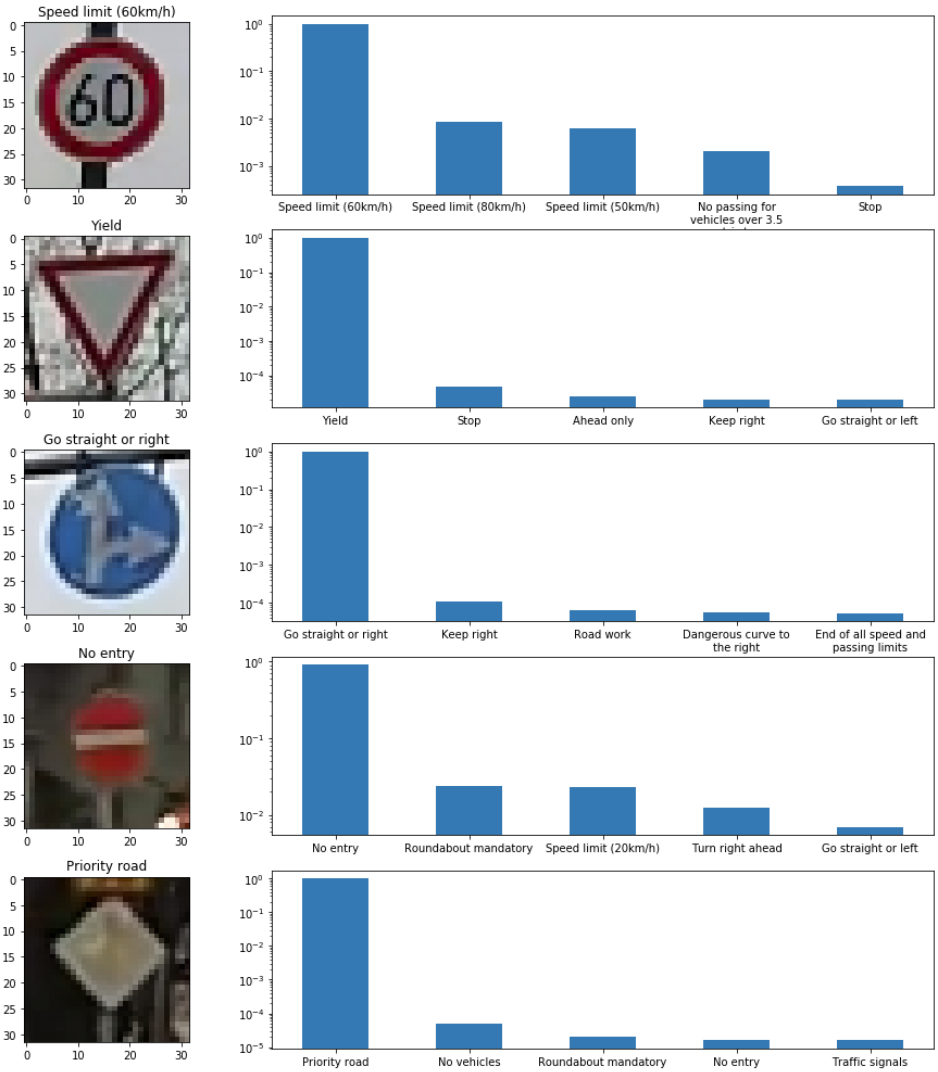
The model was able to correctly guess 5 of the 5 traffic signs, which gives an accuracy of 100%. Such a great result means that the model was trained without overfitting and could be used in the real world solutions.

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 last cell of the Ipython notebook.

For all the images model is pretty confident in result. We can see that probabilities of the result and possible alternatives differ at least by an order of magnitude. Alternatives are close to the target image by recognizible features: for example the first sign differs from the alternatives only by the number is written, and the fourth sign is similar related to form and colors used.

Results:


Speed limit (60km/h)
Speed limit (60km/h)
Speed limit (80km/h)
Speed limit (50km/h)
No passing for vehicles over 3.5
Stop
Yield
Yield
Stop
Ahead only
Keep right
Go straight or left
Go straight or right
Go straight or right
Keep right
Road work
Dangerous curve to the right
End of all speed and passing limits
No entry
No entry
Roundabout mandatory
Speed limit (20km/h)
Turn right ahead
Go straight or left
Priority road
Priority road
No vehicles
Roundabout mandatory
No entry
Traffic signals