Project report

Liang Zhang

30.09.2017

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

Rubric Points

1. Load the data set

I used *open(path_to_file, option)* function to load the train, validation and test data set.

2. Explore, summarize and visualize the data set

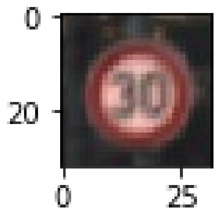
a.) Basic Summary of the Data Set

I used the built-in python functions *len()* and *set()* to calculate summary statistics of the traffic signs data set:

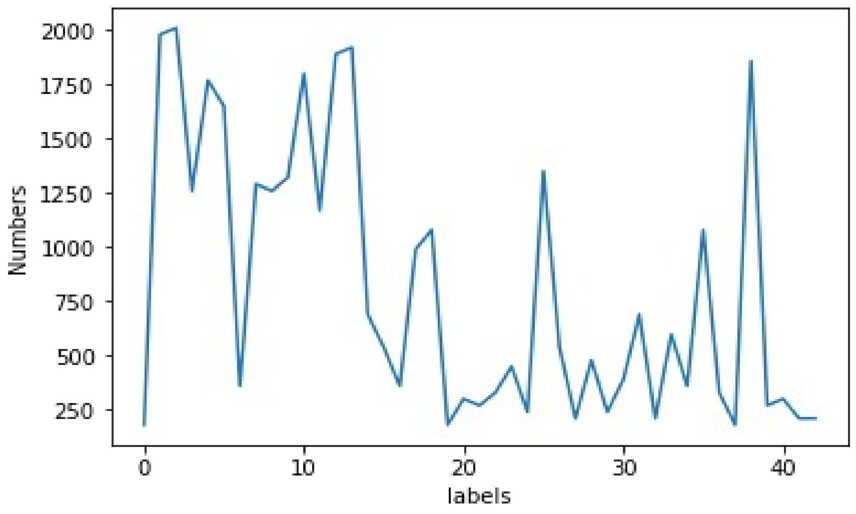
- The size of the training set is 34799.
- The size of the validation set is 4410.
- The size of the test set is 12630.
- The shape of a traffic sign image is 32x32 with 3 channels.
- The number of unique classes/labels in the data set is 43.

b.) Exploratory visualization of the dataset

A random traffic sign is shown below.



Here is an exploratory visualization of the training data set. It is a line plot of counts with respect to the labels of traffic signs.



3. Design, train and test a model architecture

a.) Preprocessing

I used the formula `(pixel - 128)/128` to approximately normalize the data. Normalizing data reduces the conditioning (the numerical error) of the problem.

b.) Model architecture

I used the LeNet architecture with an additional dropout layer. I also changed the dimensions of the inputs and outputs. The input data has dimension 32x32x3, and the output has the dimension 43.

Layer	Description
Input	32x32x3 RGB 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
Flatten	Output = 400
Fully connected	120 outpus
Relu	
Fully connected	84 outputs

Relu	
Dropout	keep_prob = 0.5
Fully connected	logits, 43 output

c.) Train the model

To train the model, I used *Epochs = 8*, *Batch_size = 32*, and *learning_rate = 0.001*. For the optimization part, I used the Adam optimization algorithm.

d.) Tuning the model

My final model results were:

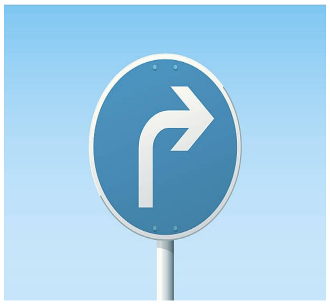
- training set accuracy of 99.5%
- validation set accuracy of 94.7%
- test set accuracy of 93.3%

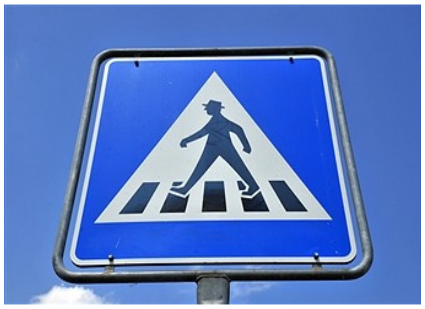
I began with the LeNet architecture, because from the exercise Lab with the traffic sign data this architecture already had a validation accuracy round 90%, which is a good starting point. I played around with the batch size a bit, and I found a smaller batch size like 32 works better than 64 and 128. The learning rate 0.001 works well compared to a smaller value 0.0005.

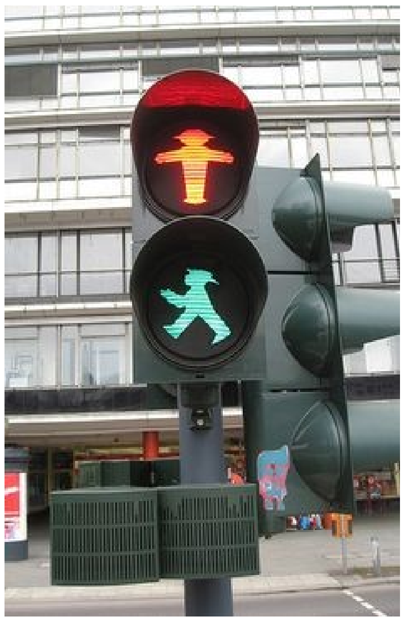
The original LeNet architecture has no regularization layer, so I assume a dropout layer should help. After I added a dropout layer, the validation accuracy improved. The model's accuracy on the training, validation and test sets are all above 93%, which in my opinion is already quite well within such a short training time.

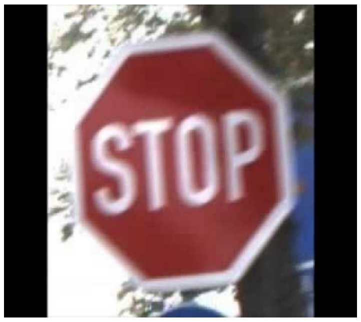
4. Test a model on new images

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









The first image (turn right ahead) is quite clear, so I expect it to be easy to classify.
The second image (pedestrians) has a lot of information in a small area, so it might be hard.
The third image (traffic signals) is a non-standard traffic signal, but is typical in Berlin. I guess it is the most difficult one.
The fourth image (stop sign) is blurred, which might be hard.
The last image (general caution) is clear and standard, so I expect it to be easy to classify.

b.) Here are the results of the prediction:

Image	Prediction
Turn right ahead	Turn right ahead
Pedestrians	Stay right
Traffic signals	Roundabout mandatory
Stop sign	Stop sign
General caution	General cation

The prediction success right is 60% with the second and the third images are incorrectly predicted.

c.) The Softmax probabilities
The top five Softmax probabilities for the first image is [99.7%, 0.2%, -, -, -]. The image is **turn right ahead**, and the model is very sure about this prediction. The model is right.

The top five Softmax probabilities for the second image is [94.3%, 3.3%, 0.9%, -, -]. The image is **pedestrians.** The model predicts **stay right** with high certainty, but the model is wrong.

The top five Softmax probabilities for the third image is [53.8%, 24.2%, 4.9%, 4.8%, 4.4%], which are **Roundabout mandatory, Road work, Keep right, Road narrows on the right and Dangerous curve to the right**, respectively. The image is a **traffic signal**, and the model is not so sure about this prediction. The model is wrong.

The top five Softmax probabilities for the forth image is [100%, -, -, -, -]. The image is **stop sign**, and the model is 100% sure about this prediction. The model is right.

The top five Softmax probabilities for the fifth image is [100%, -, -, -, -]. The image is **general caution**, and the model is 100% sure about this prediction. The model is right.

The result is semi-satisfactory. The model works well on image 1, 4, 5. But for the second image, the model gives a very certain wrong prediction. The reason might be that the resizing of the image already lost a lot information and generated noise. For the third image, the image is relative new for the model. It is not so surprise that the model can not predict it correctly.

5. Summarizing the report

With not so many lines of code with tensor flow, the modified LeNet architecture could already handle many traffic sign pictures with very high accuracy. For the outlook, I think in two areas that this program can be improved:

a.) Preprocessing

The LeNet function that I use only accept images with a prescribed size, in this case 32x32x3. Resizing the pictures and other preprocessing techniques can be improved or added.

b.) Special traffic signs

For example, the German traffic lights are common in Berlin but not so in rest of Germany. The model should be trained to handle this kind of situations. New training data can be helpful and maybe a modified model can also help for this situation.