Build a Traffic Sign Recognition Project

The goals / steps of this project are the following:

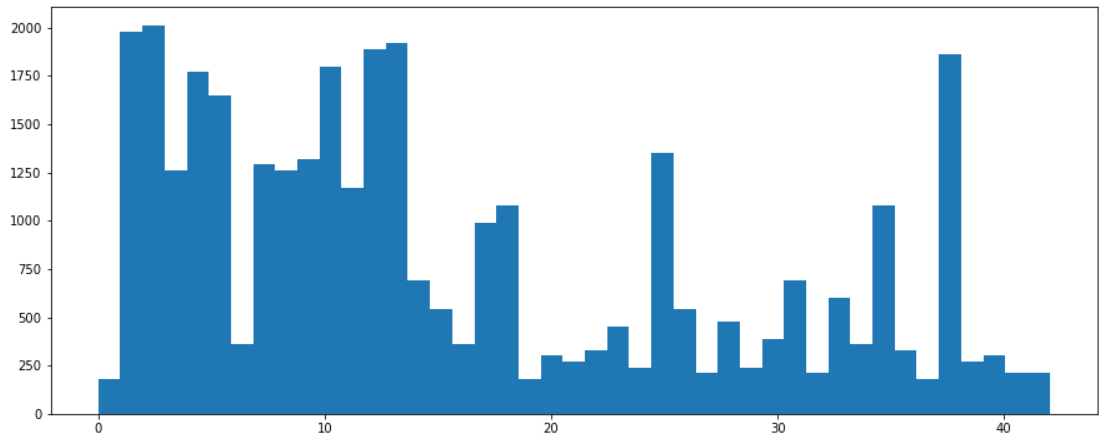
- Load the data set (see below for links to the project data set)
- Explore, summarize and visualize the data set
- Design, train and test a model architecture
- Use the model to make predictions on new images
- Analyze the softmax probabilities of the new images
- Summarize the results with a written report

1: Data Set Summary & Exploration

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.

- 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 4

Here is the label vs. examples distribution of the training data set.



2: 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?

1. I generate "new data" by just copying the old images. (I can do it better by adding a little bit of noise, rotating a few degrees). I don't think blur or upside-down transformation are good choice. Adding new data can help reducing overfitting, and make my model predict the results of the lacking images in the training set better. Now there are 203452 training examples (used to be 34799).
2. I defined some useful functions to pre-processing the images.
    - ***mm_scale(image)***: Normalized a picture, map all pixel to the range [0,1] by just dividing 255.
    - ***to_32(image)***: Convert image size： [64,64,4] -> [1,32,32,3]
3. I used *mm_scale(image)* functions to normalize all the date.

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.

4. Here is the Model Architecture for ***MyNet()***.
   Hyperparameters: mu: 0; sigma: 0.1

Layer	Description
Input	32x32x3 RGB image
Convolution1 Filter: 5x5x3x24	1x1 stride, VALID padding Output: 28x28x24
Activate	Tanh
Max pooling	2x2 stride, VALID padding Output: 14x14x24

Layer	Description
Convolution2 Filter: 5x5x24x48	1x1 stride, VALID padding Output: 10x10x48
Activate	Tanh
Max pooling	2x2 stride, VALID padding Output: 5x5x48
FC0 Input: 5*5*48	Output: 1*1200
FC1 Input: 1*1200	Output: 1x400
FC2 Input: 1*400	Output: 1x200
FC3 Input: 1*200	Output: 1x43
Activate Input: 1*43	Softmax Output: 1x43
Logits	

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.

5. Here is the table for the training Architecture

Hyperparameters	Value
rate	0.001
EPOCHS	15
BATCH-SIZE	100
Optimizer	AdamOptimizer

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

6.

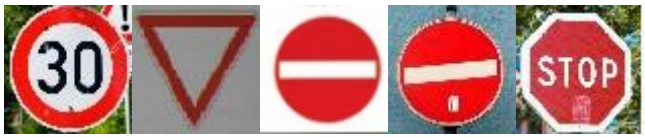
Dataset	Accuracy
Training set	0.999
Validation set	0.964
Test set	0.9493
My own 5 pictures	1.0

Summary:

I first chose the LeNet architecture, but the validation accuracy was just 0.88 after 10 Epochs. Later I let my model trained 50 epochs, the result was getting better and the validation accuracy becomes 0.94, but the it was time-consuming. Later I adjust the architecture by adding one more fully-connection layer, and make each convolutional layer deeper. I might cause overfitting. The Training Accuracy was 0.999, which was a high possibility to be overfitting. Fortunately, the Validation and Test set accuracy was 0.964 and 0.95, the performance was good enough.

3: Test the model on new images

1. Here are the 5 German traffic signs found on Google



The size for the images are [64,64,4]. Therefore, I defined the ***to_32(image)*** function to pre-processing them.

2. Here is the prediction of the 5 images

```
Prediction 14-1.jpg   : Stop                        14  It's correct
Prediction 1-1.jpg    : Speed limit (30km/h)         1  It's correct
Prediction 13-1.jpg   : Yield                       13  It's correct
Prediction 17-1.jpg   : No entry                    17  It's correct
Prediction 17-2.jpg   : No entry                    17  It's correct
```

The accuracy was 100%. It looks perfect, but I think it was because we have only 5 images. If more images were provided, I believe that the accuracy will approach to the testing accuracy.

3. Here is the table for the top5 predicitons

Images	1	2	3	4	5	real
14-1	Stop 1.0	No entry 4.27e-11	Yield 3.25e-12	30km/h 1.4e-15	59km/h 3.66e-16	Stop
1-1	30km/h 0.9974	50km/h 0.0026	End 80km/h 4.15e-7	Double Curve 5.167e-8	Wild animals 1.683e-9	30km/h
13-1	Yield 1.0	20km/h 1.221e-15	Keep left 6.536e-16	120km/h 2.215e-16	100km/h 8.501e-17	Yield
17-1	No Entry 1.0	20km/h 2.738e-10	Stop 2.348e-14	30km/h 1.267e-16	120km/h 2.806e-16	No Entry
17-2	No Entry	30km/h 4.731e-8	20km/h 3.937e-8	120km/h 5.51e-9	Stop 8.768e-11	No Entry

Here is the bar-graph. But since the other possibilities are two small, we can't even see them in the graph.