Traffic Sign Recognition

Build a Traffic Sign Recognition Project

The step of this project are the followings:

* 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

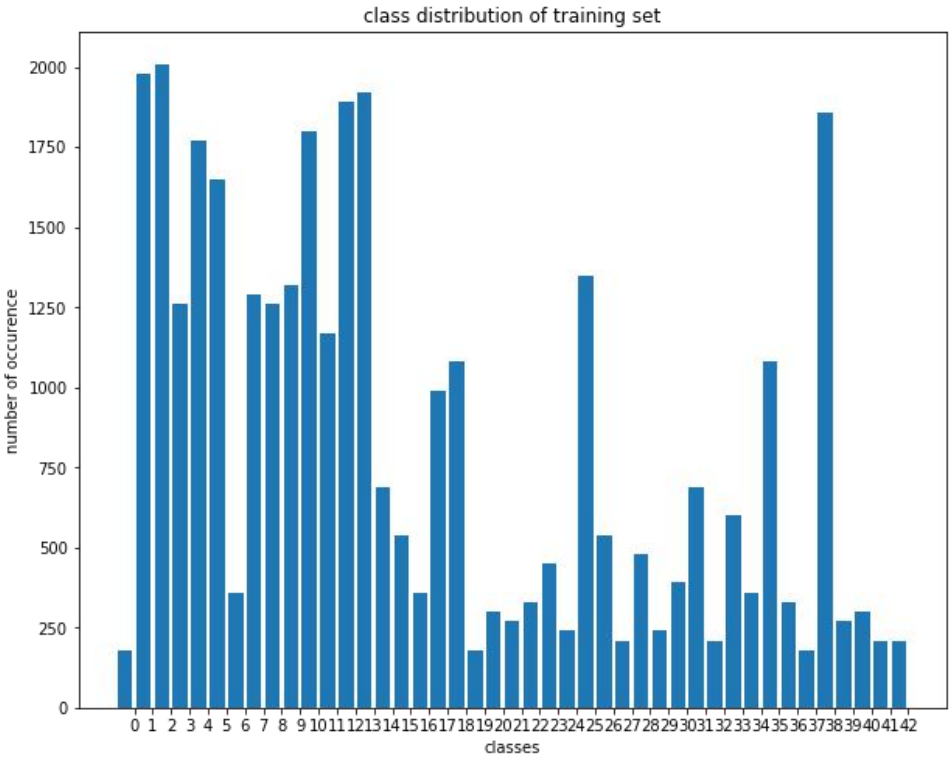
* Summarize the results with a written report

1.Dataset Summary and Visualization

I used the pandas library 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 (32,32,3)
- The number of unique labels of the dataset is 43

Here is an exploratory visualization of the data set. It is a bar chart showing how the data distributed in training set



2. Design and Test a model Architecture

I tried to use different kinds of image processing methods including gray scale ,Gaussian Blur and normalization.The best result I got for is :

(1) convert image to gray scale

(2) Scale data to [0,1]

I used the similar model architecture as the LeNet as the following layers.I applied some bit of parameters tuning.

Layer	Description
Input	32×32×1 Gray image
Convolution 5×5	1×1strides valid padding 28×28×12
Batch normalization	

Max pooling	2×2strides valid padding 14×14×12
Convolution 5×5	1×1strides valid padding 10×10×24
Batch normalization	
Max pooling	2×2strides valid padding 5×5×24
Flatten layer	600 number of units
Fully connected	400 number of units
Batch normalization	
Fully connected	120 number of units
Batch normalization	
Fully connected	84 number of units
Batch normalization	
softmax	43 number of classes

I employed AdamOptimizer to train the model.Because I used batch normalization layer to accelerate the training,the learning rate was chosen as 0.002.I set the batch size to 128 training with 30 epochs.My final model result were:

Training set accuracy of 100%

Validation set accuracy of 95.74%

Testing set accuracy of 93.59%

I firstly used the LeNet model directly to classify images and the validation accuracy and test accuracy were all around 0.92.The validation

accuracy stopped to increase after a few epochs.The LeNet model may suffer from a bit under fitting to the dataset when extracting features with convolution layers So I decided to increase the number of network parameters.I then modified the filters size to 12 and 24. One important design choice is using batch normalization layers to accelerate training steps.Experiments results show that batch normalization layers were more robust than dropout layers and achieved higher accuracy in validation and test set.

3.Test model on new images

I downloaded nine German traffic sign images from the internet.Here are those images.

All the images are shaped with 80×80×3 RGB model.The images were taken from different lighting conditions some of them are bright and some of them have dark background.I resized all these images to 32×32×1 gray scale to fit them to the model.The re-scaling step loses many pixels.All these factors may cause mis-classification of the model.

Here is the prediction result.

Ground Truth	prediction

Speed limited(70km/h)	Speed limited(70km/h)
Road work	Road work
stop	stop
Ahead only	Ahead only
Priority road	Priority road
Turn left ahead	Turn left ahead
Speed limit(60km/h)	Speed limit(60km/h)
Yield	Yield
No entry	No entry

The model successfully predicted all 9 images correctly,giving an accuracy of 100%.This compares favorably to the accuracy on the test set of 93.59%.

As observed,the model is extremely sure to its prediction.Here are the top 5 softmax probability of the first image.

probability	labels
1.0	Speed limit (70km/h)
0.0	Speed limit (20km/h)
0.0	Speed limit (30km/h)
0.0	Speed limit (50km/h)
0.0	Speed limit (60km/h)

All other images got the same softmax probabilities.The largest output in logits is much larger than other outputs so 'winner take all' happened.