PROJECT #2 Traffic Sign Classification

0, Outline

1, Project pipeline
2, Questions remaining
3, Classify results of image from web

1,The Pipe Line

Thank CarND team for providing this very good jupyter notebook in which a pipeline have been already set. I did this project simply followed this pipeline.

1) The first step is get data ready.

I learned pickle here, a package could pack data into storage or load into memory. Simple statistical summary and a random image output will give a first impression about the dataset.

Then normalize images, using x' = (x-128)/256, re-scale pixel value from [0,256] to [-1,1]. I also try add grey channel besides r,g,b, but result showed no difference. I checked the data after normalization through showing random image before and after normalize.

2) The model

According to Lenet-5, I made a five layers network describe as below:

First convolutional layer using 20 5*5(*3) filters with stride 1 and same padding. Then down sample feature map from 32*32 to 16*16.

Second convolutional layer using 40 5*5(*20) filters with stride 1 and valid padding. The down sample feature map from 12*12 to 6*6.

The second layer is also convolutional layer, 10 1*1(*40) filters combined features at the same positions of the 40 2D feature maps.

Flatten third convolutional layer outputs, generated a 360 dimensions vector.

Feeding the vector from last step to a conventional neural-nets has 120 hidden units and 43 outputs.

All activate function used here is Relu.

3) Traing and test

Using adam optimizer, set batch size and learning rate equal to 128/0.001 , after 10 epochs training. I got a 0.993 accuracy on validation set, and 0.995 on test set.

I was very surprised by the result for I could beat the #1 score of leader board.

4) Test on real data

I download origin dataset from the website, and follow the instructions finish the test. Usually the performance of 5 random selected images is 1.0, for I use the training set as test inputs, I think.

5) Showing feature maps

Just using the given code.

2, Question remaining

1) The lucky

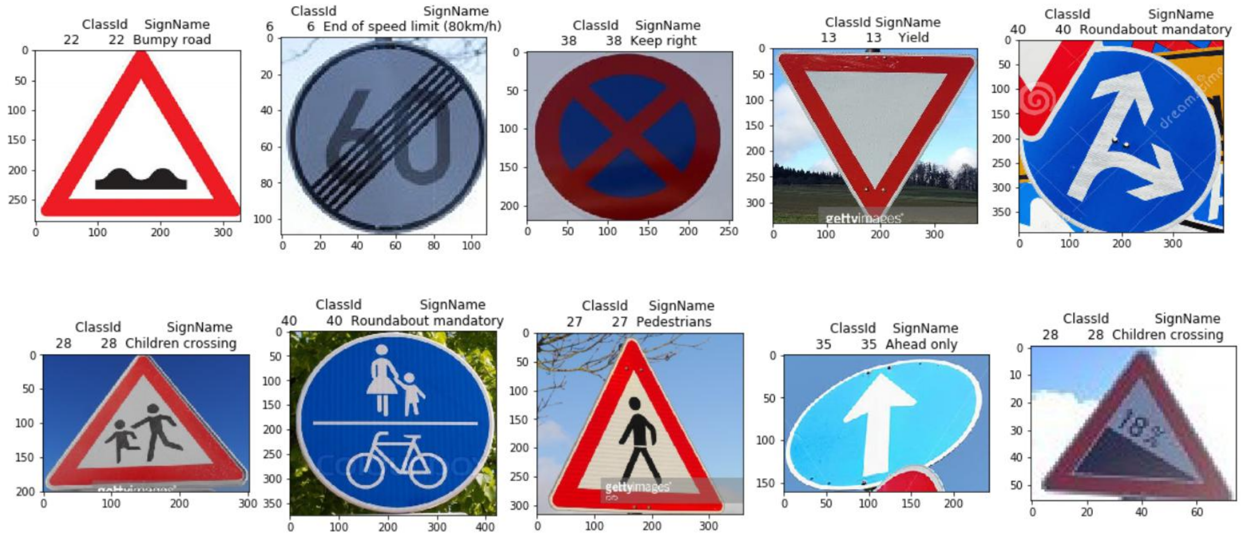
Must to say, I got a very good result by lucky. I just had a bunch of sense like making feature map small but same time make the number of feature maps large, always make features fewer than father, and using 1*1 convolution could combine features better. Then I got the state-of-art result. Totally without data augmentation and L1/L2/Dropout. Can not believe it's the first model I make, cool but strange. Is their any theory of "How to make good neural-nets"?

2) Visualization

Firstly, I made my model using a function that returns the final outputs of model. Then I found I can not reach the middle tf-Variables, not mention to visualize it. So I rewrite my model outside the function.

Is their any instructions about how to get touch to every variable in a trained model? This is a very trivial question, but perplex me a lot.

3, Classify results of image from web


ClassId SignName 22 22 Bumpy road
ClassId SignName 6 6 End of speed limit (80km/h)
ClassId SignName 38 38 Keep right
ClassId SignName 13 13 Yield
ClassId SignName 40 40 Roundabout mandatory
ClassId SignName 28 28 Children crossing
ClassId SignName 40 40 Roundabout mandatory
ClassId SignName 27 27 Pedestrians
ClassId SignName 35 35 Ahead only
ClassId SignName 28 28 Children crossing

There are 10 pics I collected from bing.com, using key words "German traffic sign". The calssify Results are shown as title of every pic.

I simple computed the accuracy on these pictures as the final result of test. I will describe every one in the order from left to right ,then up to down.

1, Bumpy road. CORRECT
2, End of speed limit(60km/h) : Wrong because we have only (80km/h) in dataset.
3, No parking. Wrong because no training data of this class
4, Yield. CORRECT
5, Mandatory direction. Wrong because no training data of this class
6, About children. CORRECT
7, Pedestrian + bicycle. Wrong because no training data of this class
8, Pedestrian. CORRECT
9, Ahead only. CORRECT
10, Steep down. Wrong because no training data of this class

For five classes that had been include in the training set, the accuracy is 100%. Compare with the test accuracy of 99.4% on GTSRB. We could conclude that little affine transformation could tolerant by model.

The other interesting thing is the "end of Speed limit". Because we do not have a (60km/h) class in training set, this sign had classified as a (80km/h) class. This result showed the model could be sensitive to small features.