Traffic Sign Classifier

Marcelo Eduardo Pederiva

Report

Data Set Summary & Exploration

In this project I read the train.p, valid.p and test.p using pickle load.

I obtained the following results:

Number of Training examples = 34799

Number of Validation examples = 4410

Number of Testing examples = 12630

Image data shape = (32, 32, 3)

Number of Classes = 43

To have a better exploration about the data set I put in a graph showing the number of images of each Class (Figure 1).

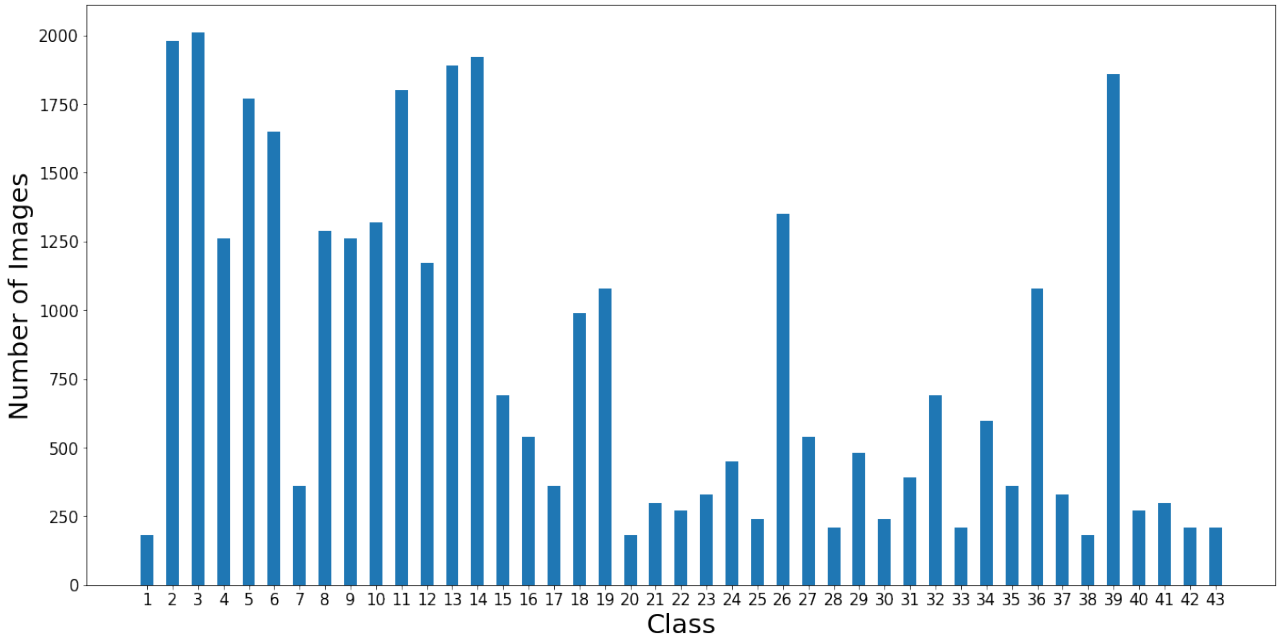

Figure 1 – Graph of the Number of images for each Class

Design and Test a Model Architecture

Pre-process the image data

To pre-process the image I transformed the RGB data in a Gray-scale to make the learning process fast and easier. After that, I normalized the image using the Equation(1) to increase the contrast of the image and make the features more identifiable (Figure 2).

$$(pixel - 128)/ 128 \qquad \textit{Equation(1)}$$

With this 2 process I obtained a better Training result.

Figure 2 -On the Left the input Image and on the Right the Gray and Normalized Image

LeNet

My final model consisted of the following layers

Input	32x32x1 Gray Image
Convolution 3x3	1x1 stringing, same padding, Output 28x28x32
RELU	
Max Pooling	2x2 stride, Output 14x14x32
Convolution 3x3	1x1 stringing, same padding, Output 10x10x64
RELU	
Max Pooling	2x2 stride, Output 5x5x32
Flatten	Output 1600
Fully Connected	Output 120
RELU	
Fully Connected	Output 84
RELU	
Fully Connected	Output 43

To train my model I used 20 Epochs, Batch size = 64 and Learning Rate = 0.0009.

With this parameters my Final Model results:

Validation Accuracy = 0.958

Test Accuracy = 0.940

Stating this project I tried changing the "mu" and "sigma" parameters from truncated_normal function, because it was parameters that I had never changed.

After the validation accuracy didn't change so much, I focused in the preprocessing the image. Changing the image to gray and normalizing it, I got a good increase in the accuracy, but not enough. So I changed the weights and bias increasing the output depth. The sign usually have similar and linear shapes, with more filters, maybe it was easier to distinguish and identify each one. With this thought I increase then and got a validation accuracy > 0.93. I changed the batch size, but didn't had greats improvements. To complete, I decrease the learning rate because my accuracy training was oscillating too much. As the result, I got validation accuracy>0.95.

I believe that my project got good results to the implementation of the learning traffic signs, but still need to spend more time studying the parameters, increasing the number of layers (pooling, dropout, convolution, etc) and correcting the perspective and colors from the input images.

The sign learning it's essential to create a self drive car that identify how to follow the Traffic Rules and drive safety in the street between another cars. The learning process need to have an increase accuracy because when we talk about traffic sign, for example, the program can't confuse a "Stop" sign from an "Ahead Only" sign.

Test a Model on New Images

To test my Model, I enter with 10 German Traffic Signs that I had founded on the web and put the program to identify them.

The program Results:

Image	Prediction
Stop Sign	Stop Sign
Speed Limit (70 km/h)	Speed Limit (70 km/h)
Turn Right Ahead	Turn Right Ahead
Priority Road	Priority Road
Yield	Yield
Go Straight or Left	Priority Road
End of no Passing	End of no Passing
No Entry	No Entry
No Vehicles	No Vehicles
Bumpy Road	Bumpy Road

My model got right 9 of 10 images, making the accuracy of 90%.

Comparing the accuracy of the test with the web images, we can observe that the test got an increase accuracy. The resize function, distortion and different background from the web images maybe make hard to identify what sign it is. Furthermore, I only used 10 images to test the accuracy, with more inputs I could find a similar accuracy from the test.p images.

Looking the Top 5 Softmax probabilities for each prediction we have:

All the signs that the program got right had nearly 100%, the lowest certain of what sign it is occurs in the wrong prediction, with about 70%.

Stop Sign:

Stop: 99.99%
Speed limit (60km/h): 0.00%
Speed limit (50km/h): 0.00%
Go straight or right: 0.00%
Keep right: 0.00%

Speed Limit (70 km/h):

Speed limit (70km/h): 100.00%
Speed limit (20km/h): 0.00%
Speed limit (30km/h): 0.00%
General caution: 0.00%
Speed limit (80km/h): 0.00%

Turn Right Ahead:

Turn right ahead: 100.00%
Ahead only: 0.00%
Go straight or left: 0.00%
Keep left: 0.00%
Road work: 0.00%

Priority Road:

Priority road: 100.00%
Roundabout mandatory: 0.00%
No entry: 0.00%
End of all speed and passing limits: 0.00%
Yield: 0.00%

Yield:

Yield: 100.00%
Children crossing: 0.00%
Speed limit (30km/h): 0.00%
Beware of ice/snow: 0.00%
Speed limit (50km/h): 0.00%

Go Straight or Left:

Priority road: 71.11%
No entry: 23.84%
Road work: 2.41%
Traffic signals: 2.29%
Beware of ice/snow: 0.26%

End of no Passing:

End of no passing: 99.92%
Dangerous curve to the right: 0.05%
No passing: 0.02%
Go straight or right: 0.01%
End of all speed and passing limits: 0.00%

No Entry:

No entry: 100.00%
Stop: 0.00%
Turn right ahead: 0.00%
No passing: 0.00%
Ahead only: 0.00%

No Vehicles:

No vehicles: 100.00%
Traffic signals: 0.00%
Yield: 0.00%
Priority road: 0.00%
Speed limit (30km/h): 0.00%

Bumpy Road:

Bumpy road: 100.00%
Keep right: 0.00%
Road narrows on the right: 0.00%
Road work: 0.00%
Traffic signals: 0.00%