

Write-up Report – Traffic Sign Classifier Project

Self-Driving Car Nano Degree Program – Term1, Project2

1. Objective

The objective of this project is to model a deep neural network that will predict the German traffic signs. Following are the steps of this project.

- 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

2. Rubric Points

2.1. Write-up / README

- 2.1.1. Provide a Write-up / README that includes all the rubric points and how you addressed each one. You can submit your write up as markdown or pdf. You can use this template as a guide for writing the report. The submission includes the project code.

This document is the write-up report. The link to the project code is

https://github.com/sekar1232007/Traffic_Sign_Classification

2.2. Data Set Summary & Exploration

- 2.2.1. 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

I used basic python functions such as len, shape to find the length & dimensions of the training and test data set. I also wrote a small code to read in the signnames.csv file to find the no. of classes and to import the names of each traffic signs.

Note: this information can be used later to print the name of the sign after predicting it.

Following is the summary of the data set.

- Number of training examples = 34799
- Number of testing examples = 4410
- Image data shape = (32, 32, 3)
- Number of classes = 43

2.3. Design and Test a Model Architecture

2.3.1. Data preprocessing

- I decided not to convert the image to grayscale, as the color information may play a significant role in differentiating similar signs.
Ex: there are minimum and maximum speed limit signs. Both the signs have numbers, but the background colors are different in both the cases.
- I plotted a random image to check it visually. This is how it looked like



- I normalized the input data between 0 and 0.9. This reduces the range of input values and as a result will help in converging (reaching local minima) weights and biases in a stable way.

2.3.2. Model Architecture

My final model architecture consists of following layers

Layer	Input Size	Output Size	Additional Info
Input	32*32*3	N.A	Input image (R.G.B values)
Convolution	32*32*3	28*28*18	Filter =5*5, strides=1*1, Valid padding
ReLu activation	N.A	N.A	N.A
Max pooling	28*28*18	14*14*18	Filter=2*2 strides=1*1
Convolution	14*14*18	10*10*54	Filter =5*5, strides=1*1, Valid padding
ReLu activation	N.A	N.A	N.A
Max pooling	10*10*18	5*5*54	Filter=2*2 strides=1*1
Flatten	5*5*54	1350	
Fully connected	1350	500	
Fully connected	500	250	
Output layer	250	43	

2.3.3. Model Training

At first I reused the Lenet model architecture from character recognition lab. During training the accuracy was very low. After debugging it was very clear that the model was under fitting. This model was designed only to predict 10 classes and input feature was only one dimension. Where as in traffic recognition, I used an input with 3 features and there are 43 classes to predict.

I tried increasing the features of my convolutions layer, after some iterations I could achieve 93% of accuracy on the training set with 18 & 54 features respectively at convolution layers 1 & 2.

2.3.4. Parameters used

Batch size=128

Learning rate = 0.001

Weights and Biases initialized with a random normal distribution with $\mu=0$ & $\Sigma=0.1$

Epochs =10

2.3.5. Accuracy

Validation: 94.6%

Test: 92.4%

2.3.6. Summary

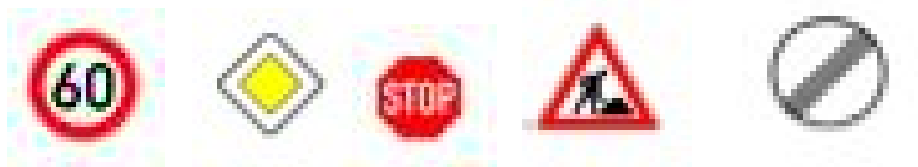
The model reused a Lenet architecture with a convolution->pooling->activation layers. Following are the justification for this architecture.

- Convolution Layer are good for image classification problem as they look for various features with increasing complexity at higher layers. Since, this problem is classification of traffic signs based on images, the convolution method is an optimal way to approach
- A max pooling layer is used after the convolution as it reduces the size of the feature map without losing much valuable information. This also reduces the no. of parameters required in the next layer, thus reducing the computational speed

- A ReLu activation layer introduces non-linearity to the overall model architecture which helps in better regularization of the model. Also, it avoids the vanishing gradient problem associated with the sigmoid function.
- Dropout was not used as the was well generalized & the accuracy has already reached above 93% and this reduced the complexity of the model

2.4. Test a Model on New Images

2.4.1. German Traffic sign used for testing



2.4.2. Image resizing

After downloading, I resized the images to 32*32 manually using paint brush application and renamed the file names with their corresponding labels

2.4.3. Image prediction

- At first, all the images were predicted wrongly by my model. After a long debugging session, I found the reason for wrong prediction.
- Reason:
I used imread function to read the png files and then normalized them using the same algorithm used for training images.
- The imread function has already normalized the data between 0 & 1. This has led to the double normalization and all the input values scaled almost close to 0.
- Once I corrected the double normalization, the model predicted all the 5 images correctly. The accuracy was 100%

2.4.4. Comparison with test data set

The accuracy on the captured images is 100% whereas 93% on testing set. This may be because of very less number & decent quality images are used for prediction & accuracy calculation. When more complex images (Ex: signs

captured @ different angles, signs captured @ night time or foggy images) are used, Then the accuracy may be lesser than 100%.

2.4.5. Top 5 softmax probabilities

Almost 100% for all the five images.

The model predicts the first 4 images with 100% certainty, whereas the 5th image i.e. “Class id 32: End of all speed and passing limits” with 51%.

Following is the probable reason:

While classifying the 5th image, the second highest probability is 49.3% (“class id:6, End of speed limit (80km/h)”). Both the images look almost similar and the model is trained with very less examples i.e. (210 for class 32 & 330 for class 6).

The network need to be trained with more examples for these two images to identify the subtle differences between them and to predict with high certainty

```
TopKV2(values=array([[ 1.00000000e+00,  1.46327493e-21,  3.05450887e-23,
 7.43973458e-24,  2.94495912e-27],
 [ 1.00000000e+00,  1.08557615e-16,  2.50128416e-17,
 1.05531024e-19,  6.87636665e-20],
 [ 1.00000000e+00,  1.94675594e-15,  4.83798543e-18,
 6.49346959e-21,  2.72215300e-22],
 [ 1.00000000e+00,  9.08220493e-13,  1.89497750e-13,
 2.09174249e-18,  1.10104791e-18],
 [ 5.06199956e-01,  4.93799031e-01,  1.00174452e-06,
 7.87055754e-10,  2.73833300e-12]], dtype=float32), indices=array([[ 3,  2,  5, 29,  1],
 [12, 26, 11, 30, 29],
 [14, 17,  8,  1,  5],
 [25, 31, 30, 14, 24],
 [32,  6, 41,  5,  3]]))
```