Autonomous Driving Project # 3 – Traffic Sign Classifier

## Himanshu Kumar

# Project Definition:

This project has following requirements:

* 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

Below section will explain each point in detail.

# Loading Data Set:

Data set was already saved as a pickle file and we load it using python pickle module. The data set were already categorized in 3 set:

1. Training – The images and labels in this category are used for training CNN.

2. Validation – The images and labels in this category are used to evaluate our training model.

3. Test – The images and labels in this category are finally used for prediction by our model and calculating its accuracy.

# Exploring Data Set:

I did some exploration of the data set to get idea about some of the stats, which are below:

Number of training examples = 34799

Number of testing examples = 12630

Number of validation examples = 4410

Image data shape = (32, 32, 3)

Number of classes = 43

5 Sign with highest Samples: [38, 12, 13, 1, 2]

5 Sign with lowest Samples: [0, 37, 19, 32]

I also plotted some of the stats for the dataset. Some of them are below:

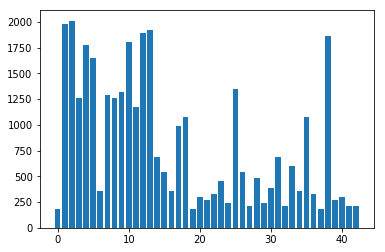


Figure 1: Class Distribution in Training Data Set

One of the traffic sign with lowest number of samples in training data set:



Figure 2: Traffic sign with lowest number of Samples

One of the traffic sign with high number of samples in training data set:



Figure 3: Traffic sign with highest number of Samples

# Design/Training/testing on Data Set:

Training pipeline:

1. First, I did some pre-processing on the images in the dataset. For pre-processing, I used

a) Converting images from RGB to gray scale. I did this conversion because of 2 reasons, 1) color information is not useful for image classification and sometime might introduce noise because of lighting condition and 2) Training on 1 channel is faster than training on 3 -channel

b) Then, I normalized images by using:

(pixel – 128)/128

The idea behind normalization is to get 0 mean with variance of 1. This make image pixel intensity to follow normal distribution and also helps in removing noise.

// put image here before and after

2. Then, I used LeNet architecture from previous lesson. I had to change number of output class from 10 to 43 and also I added dropout with keep probability of 0.7 for training and 1.0 for validation/testing.

Adding dropout in the LeNet did help me in increasing model accuracy on validation data set. Dropout makes model not to rely on any single output or develop dependency on output and thus prevented the model from overfitting.

|  |  |
| --- | --- |
| Layer | Description |
| Input | 32x32x1 Gray scale image |
| Convolution 1 | filter size = 5, Stride = 1, padding = Valid and number of filter is 6, input channel = 1. This gives output of 28x28x6 |
| Activation | RELU |
| Max Pooling | filter size = 2, Stride = 2, padding = 0. This gives output of 14x14x6 |
| Convolution 2 | filter size = 5, Stride = 1, padding = Valid and number of filter is 16, input channel = 6. This gives output of 10x10x16 |
| Activation | RELU |
| Max Pooling | filter size = 2, Stride = 2, padding = 0. This gives output of 5x5x16 |
| Flatten | Output = 400 |
| Fully Connected | Input is 400 and output is 120 |
| Activation | RELU |
| Regularization | Dropout with keep probability of 0.7 for training and 1.0 for validation and testing |
| Fully Connected | Input is 120 and output is 84 |
| Activation | RELU |
| Regularization | Dropout with keep probability of 0.7 for training and 1.0 for validation and testing |
| Fully Connected | Input is 84 and output is 43 (in logits) |

3. Some of the hyper parameter for training:

Epochs: 11

Batch Size: 128

Learning Rate: 0.001

4. Then using tensor flow API, we calculate cross-entropy

cross\_entropy = tf.nn.softmax\_cross\_entropy\_with\_logits(labels=one\_hot\_y, logits=logits)

which is used by Adam Optimizer to reduce the error and rebalance weights during back propagation.

5. I got accuracy of 94.7% on validation data set using my trained model. For test dataset, the accuracy was 92.3%.

# Prediction on New Image:

On image downloaded from web, I got accuracy of 20%. The accuracy is very poor in this case which may be due to following reasons:

1. Images were downloaded randomly from the web and had to be go through resizing. Resizing might have caused in loss of some of the information.

2. Most of the images 4 out of 5 has a smaller number of representations in training dataset which might have also affected accuracy.

The images are in directory: CarNDTrafficSignClassifierProject/GermanDownloadImage

Images are:

|  |  |  |
| --- | --- | --- |
| Image | Class ID | # Samples in Training |
| Dangerous curve left | 19 | 180 |
| Dangerous curve right | 20 | 300 |
| Narrow Road Right | 24 | 240 |
| Ice Road | 30 | 390 |
| Priority Road | 12 | 1890 |



Figure 4: Ice Road



Figure 5: Dangerous Curve Right



Figure 6: Narrow Road Right



Figure 7: Dangerous Curve Left



Figure 8: Priority Road

Here is softmax probability for 5 images:

|  |  |
| --- | --- |
| Images | Prediction |
| Dangerous curve left |  |
| Dangerous curve right |  |
| Narrow Road Right |  |
| Ice Road |  |
| Priority Road |  |