

German Traffic Sign Classifier – CarND assignment

Alberto Escarlate¹

¹Affiliation not available

August 22, 2017

Abstract

This is the submission for the Traffic Sign Classifier project part of Udacity Self-Drive Engineer Nanodegree. The assignment was to train a Convolutional Neural Network (CNN) model to classify traffic signs from the [German Traffic Sign Dataset](#).

Using a model based on the LeNet architecture enabled my classifier to achieve the required 93% training accuracy.

1 Goals

The goals of the project were:

- 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

2 Dataset Exploration

2.1 Data structure

The data, from the [German Traffic Sign](#) dataset, is a pickled dictionary with 4 key/value pairs:

- **features:** 4D array containing raw pixel data of the traffic sign images, (num examples, width, height, channels).
- **labels:** 1D array containing the label/class id of the traffic sign. The file `signnames.csv` contains id -> name mappings for each id.
- **sizes:** list containing tuples, (width, height) representing the original width and height the image.
- **coords:** list containing tuples, (x1, y1, x2, y2) representing coordinates of a bounding box around the sign in the image.

2.2 Dataset contents

The dataset provided for this assignment already contained a subset of images cropped and resized to 32 x 32 pixels so we had to consider only **features** and **labels**.

The dataset provided for this assignment already contained a subset of images cropped and resized to 32 x 32 pixels so we had to consider only **features** and **labels**.

Examining the data we could find the following numbers:

- Number of training examples: 34,799
- Number of testing examples: 12,630
- Image data shape: (32, 32, 3)
- Number of classes: 43

2.3 Classes

Another file provided was a list of classes with their respective names. Here we display 5 rows randomly picked.

ClassId	SignName
11	Right-of-way at the next intersection
25	Road work
31	Wild animals crossing
39	Keep left
40	Roundabout mandatory

Figure 1: This is a caption

And here are some training images for each of the 43 classes.



Figure 2: Display of all classes with a sample training image.

Inspecting the training dataset we notice, in the histogram image below, that the classes aren't equally balanced. With a considerable number of classes having less than 500 samples the model can have its testing accuracy impacted.

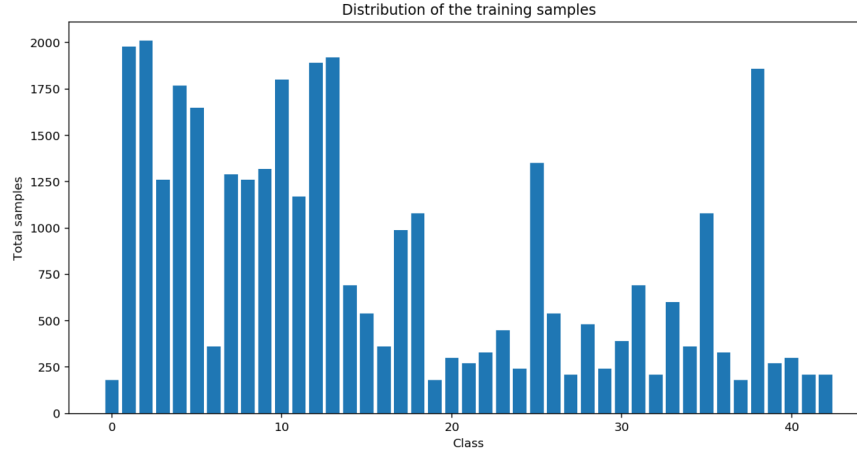


Figure 3: Distribution of the training samples

3 Model Architecture

3.1 CarND model vs. LeNet

The model architecture I implemented *CarND* was similar to the the *LeNet* architecture ([LeCun et al., 1998](#)) with a slight modification. I included dropout layers after each of the Fully Connected layers and avoid overfitting.

CarND model		Layer	LeNet model	
Input	Output		Input	Output
32x32x3	28x28x6	Convolution + ReLU	32x32x3	28x28x6
28x28x6	14x14x6	Pooling	28x28x6	14x14x6
14x14x6	10x10x16	Convolution + ReLU	14x14x6	10x10x16
10x10x16	5x5x16	Pooling	10x10x16	5x5x16
5x5x16	400	Flatten	5x5x16	400
400	120	Fully Connected + ReLU	400	120
prob = 0.5		Dropout	n/a	n/a
120	84	Fully Connected + ReLU	120	84
prob = 0.5		Dropout	n/a	n/a
84	43	Fully Connected	84	10

Figure 4: Comparing CarND model with LeNet

As a last step, I implemented a few image processing techniques and test whether these would make the model increase its accuracy. The techniques were:

- Image normalization
- Histogram equalization
- Gaussian blur
- Image sharpening
- Downsampling to grayscale

Overall the transformations didn't change the accuracy significantly, so I decided to proceed with only downsampling the images to grayscale. That was enough to get to the required 93% accuracy.

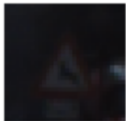
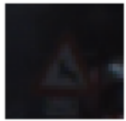







Augmentation technique	Validation accuracy (40 epochs)	Sample image
No augmentation	0.923	Wild animals crossing 
normalize	0.900	Wild animals crossing 
equalize_histogram	0.949	Wild animals crossing 
gaussian_blur	0.911	Wild animals crossing 
sharpen	0.926	Wild animals crossing 
grayscale	0.942	Wild animals crossing 
equalize_histogram, grayscale	0.937	Wild animals crossing 
equalize_histogram, sharpen	0.822	Wild animals crossing 
all together	0.935	Wild animals crossing 

Figure 5: Comparing validation accuracy when using image transformations in the data set

Starting with LeNet already put me in a good position to iterate the model and adjust the hyperparameters. From previous experience in the **Deep Learning Foundation nanodegree** I knew I should play with

adding new layers and dropout after the fully connected layers. Then I had to iterate the hyperparameters to fine tune the results. One hyperparameter that got me stuck for some time was the batch size, I assumed that a large batch would work better and only when I decreased from 256 to 128 and then to 64 that I got favorable accuracies.

3.2 Hyperparameters

- Epochs = 100
- Batch size = 64
- Dropout keep probability = 0.50
- Learning rate = 0.0005

3.2.1 Weight initialization

- $\mu = 0$
- $\sigma = 0.1$

3.3 Achieved Accuracy

- Validation Accuracy = 0.942
- Test Accuracy = 0.925
- Training Accuracy = 1.000

4 Testing the model with new images

Once the model was stable, I found a set of images of German traffic signs on the Internet to test how it would predict the traffic sign classes.



Figure 6: Signs found on the Internet

The signs downloaded from the Internet (figure 6) were then tested against the trained model. Running the prediction had an accuracy of 0.833. The images I chose are likely to have better results as they are well centered and have no background that could mislead the model. The only prediction that failed to predict was very similar to the actual image (a triangular sign with red border and white interior).

Calculating the top 5 Softmax probabilities for each image the model strongly predicted the top choices with probabilities getting almost all close to 100%. The model accuracy in the small set of five samples got to 83.3% compared to 92.5% for the test set accuracy.

In the test showed above the two incorrect predictions had the correct class in the top 5. Besides the top choice is quite similar visually to the correct class.

Expected 28 -> predicted 38
Expected 17 -> predicted 17
Expected 12 -> predicted 12
Expected 25 -> predicted 25
Expected 14 -> predicted 14
Expected 10 -> predicted 10

✓ Correct predictions: 5 out of 6 , accuracy = 0.833

Figure 7: Prediction for images found on the Internet



Figure 8: First column shows the correct class. Columns 2-6 show the predicted class sorted by probability.

References

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.