CarND Traffic Sign Classifier Project Writeup

Tiffany Huang

November 4, 2017

1 Traffic Sign Recognition Project

This project consists on the following steps/tasks:

- 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.
- Summarize the results in a written report.

My project code can be found here:

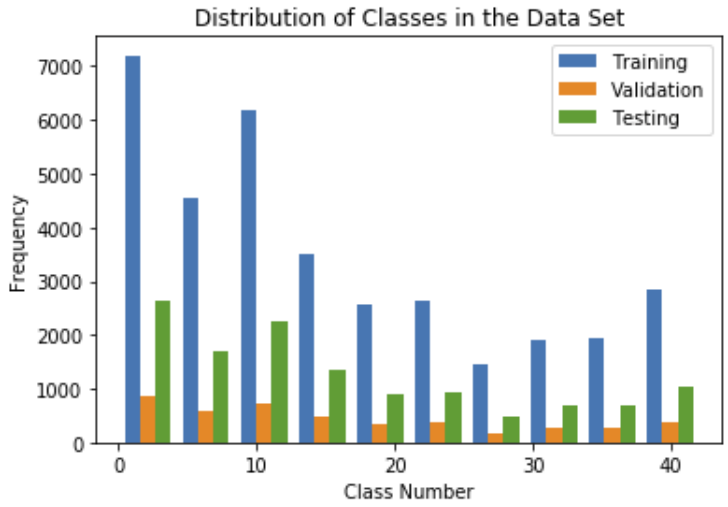
2 Data Set Summary & Exploration

I used the `pandas` library to provide a summary of the data set. I found that the

- Number of training examples = 34799
- Number of validation examples = 4410
- Number of testing examples = 12630
- The shape of a traffic sign image was (26, 25)
- Number of unique classes = 43

3 Exploratory Visualization of the Dataset

To visualize the data, I used `matplotlib` to show the distribution of classes in the training, validation, and testing set. The histogram shows how many examples of each class exist in each dataset.

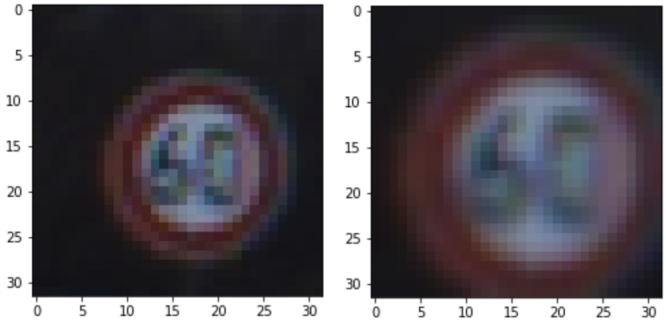


4 Design and Test of a Model Architecture

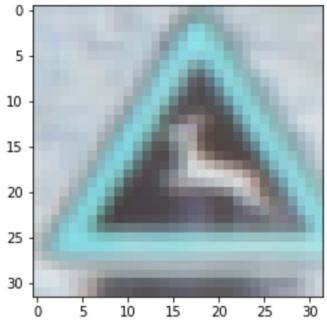
4.1 Preprocessing

The first step in my preprocessing was to crop the image so that only the traffic sign is in the image. This step removes the background of the image, which in most cases does not provide any useful information to determine what kind of sign is in the image. For instance, grass does not help to indicate a stop sign vs. a general caution sign. By removing the excess information-lacking data allows the CNN to focus on the important features in the image. To do the cropping correctly, I had to resize the image to the original size because the bounding boxes for the signs were given with respect to the original image size. After the cropping, I then resized the image to 32 $\times$ 32 $\times$ 3 because my CNN takes this size image as input. An example is shown below:

The next step was to normalize the image because different images were taken with different lighting, but different lighting should not affect the



learned results. Thus, normalizing the image by subtracting the mean of the image and dividing by the standard deviation allows us to account for these environmental condition differences between images. An example of normalization is shown below:



I chose not to convert the images to grayscale because I thought the colors of the traffic signs would be important features for the model to use for prediction.

4.2 Final Model Architecture

My final model consists of the following layers:

Layer	Description
Input	32 x 32 x 3 RGB Image
Convolutional	1 x 1 stride, VALID padding, outputs 28 x 28 x 6
Activation	ReLU
Max pooling	2 x 2 window, 2 x 2 stride, SAME padding, outputs 14 x 14 x 6
Convolutional	1 x 1 stride, VALID padding, outputs 10 x 10 x 16
Activation	ReLU
Max pooling	2 x 2 window, 2 x 2 stride, SAME padding, outputs 5 x 5 x 16
Flatten	Outputs 400 x 1
Fully connected	Outputs 120 x 1
Activation	ReLU
Fully connected	Outputs 84 x 1
Activation	ReLU
Fully connected	Outputs 43 x 1

5 Testing a Model on New Images

6 Visualizing the Neural Network