CarND Traffic Sign Classifier Project Writeup

Tiffany Huang

November 15, 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: `https://github.com/tahuang/Traffic_Sign_Classifier`

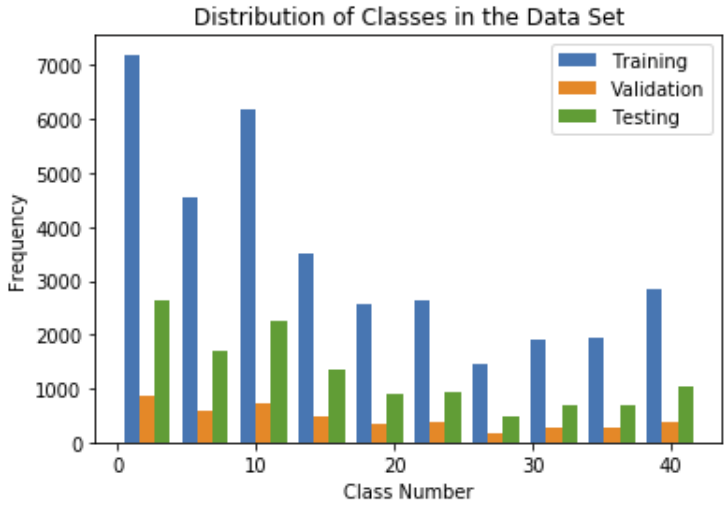
2 Data Set Summary & Exploration

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 the first traffic sign image in the training set was (26, 25)
- Number of unique classes = 43

3 Exploratory Visualization of the Dataset

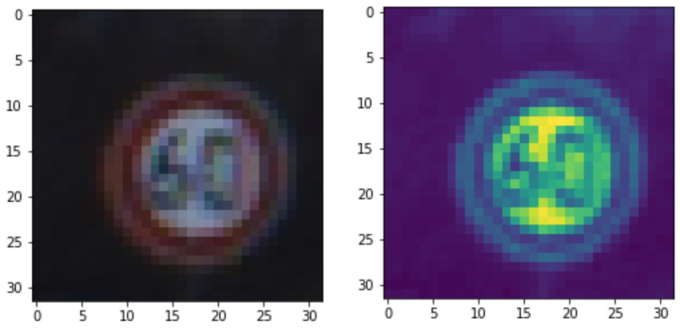
To visualize the data, I used `matplotlib` to show the distribution of classes in the training, validation, and testing set. The histogram shows how many examples of each class exist in each dataset.



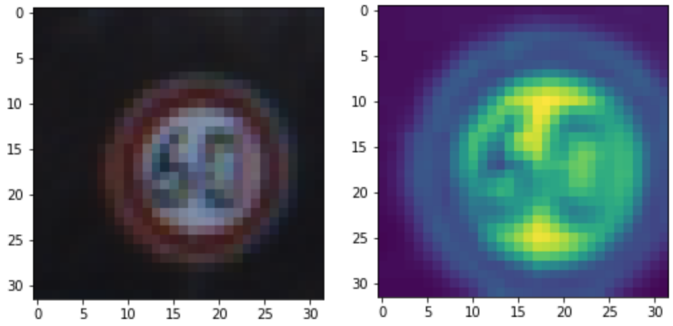
4 Design and Test of a Model Architecture

4.1 Preprocessing

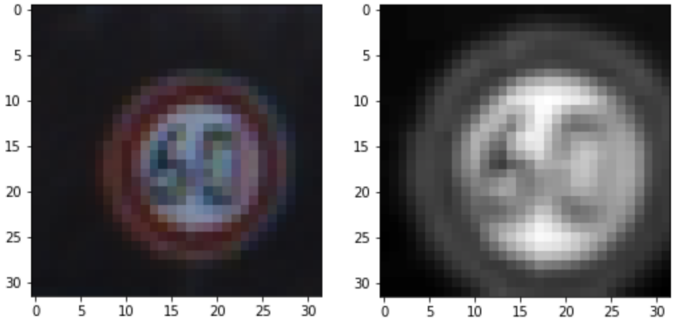
My preprocessing code is in the 5th cell of the Ipython notebook. The first step in my preprocessing was to grayscale the image (line 10 in the 5th cell) because although colors can help determine the class of traffic sign, the shapes and gradients on the image are more reliable features that aren't subject to different lighting conditions. Below is an example of before and after grayscaling. The color is a bit off, but it's fixed after the normalization step at the end.



Next, I cropped the image (lines 12 -19) 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. After removing the excess irrelevant data, the CNN can 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 32×32 because my CNN takes this size image as input. An example is shown below:



The next step was to normalize the image (line 23) 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. I then layered this image into a 3 channel image with the same grayscale image in each channel (line 24) to accommodate the input requirement of my CNN, which is a $32 \times 32 \times 3$ image. An example of normalization is shown below:



4.2 Final Model Architecture

My final model is outlined in the 6th cell of the Ipython notebook and consists of the following layers:

Layer	Description
Input	32 x 32 x 3 RGB Image
Convolutional	1 x 1 stride, VALID padding, outputs 30 x 30 x 4
Activation	ReLU
Convolutional	1 x 1 stride, VALID padding, outputs 29 x 29 x 5
Activation	ReLU
Convolutional	1 x 1 stride, VALID padding, outputs 28 x 28 x 6
Activation	ReLU
Average 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
Average 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 300 x 1
Activation	ReLU
Dropout	Keep probability 0.8 (training only, keep probability 1 for validation and testing)
Fully Connected	Outputs 200 x 1
Activation	ReLU
Dropout	Keep probability 0.8 (training only, keep probability 1 for validation and testing)
Fully Connected	Outputs 120 x 1
Activation	ReLU
Dropout	Keep probability 0.8 (training only, keep probability 1 for validation and testing)
Fully Connected	Outputs 84 x 1
Activation	ReLU
Dropout	Keep probability 0.8 (training only, keep probability 1 for validation and testing)
Fully Connected	Outputs 43 x 1

To train the model, I used a batch size of 128 with 10 epochs (cell 6) and an Adam optimizer with a learning rate of 0.001 (cell 8). The optimizer was trying to optimize the softmax cross entropy between the logits from the CNN and the one-hot encoded labels provided by the training set.

My final model results were a training accuracy of 0.975, a validation accuracy of 0.947, and a test set accuracy of 0.917. This is calculated in the `evaluate(X_data, y_data, dropout_keep)` function which is called in cell 10. The test accuracy is calculated in the "Predict the Sign Type for Each Image" section in cell 13.

I chose to use the LeNet architecture [1] because it was originally used to classify numbers in an image. Since, traffic signs can also be classified based on similar features to numbers such as shapes (and some traffic signs even have numbers on them), I believed LeNet to be a good starting point for the traffic sign classification problem.

LeNet by itself could only achieve a validation accuracy of about 0.880, so I had to adjust the architecture a bit. I had an overfitting problem because the training accuracy was about 0.1 higher than the validation accuracy. I added two convolutional layers on top of LeNet because traffic signs are generally more complicated than simple numeric digits. Additional convolutional layers allow the network to build up a more complex understanding of the image, combining simpler characteristics in earlier layers like basic lines and simple shapes to form a traffic sign.

I changed the max pooling in the LeNet to average pooling because it seemed like max pooling could have been too much of a loss of information and average pooling could keep more of the data in the image while still preventing overfitting.

I also added one more fully connected layer and dropout layers after 3 of the 4 fully connected layers. Dropouts prevent overfitting because they randomly drop units from the network during training so that the final model isn't so heavily dependent on the training data. I tuned the keep unit probability for the dropout layers empirically by just testing a few to see which one achieved the best validation performance. I found that a keep probability of 0.8 worked the best during training.

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proc. IEEE*, vol.86, no. 11, pp. 2278-2324, Nov. 1998.

5 Testing a Model on New Images

To test my model on brand new images, I found these five German traffic signs on the web:

(a) Image 1

(b) Image 2

(c) Image 3

(d) Image 4

(e) Image 5

The second and third images might be a little difficult to classify because they have various shading across the traffic sign. Additionally, the second and fourth images might be challenging because they have irrelevant signs near the main traffic sign. However, using normalization and cropping, these problems should be mitigated through preprocessing. Here are the results of the prediction of my model, which was done in cell 13 line 21:

Image	**Prediction**
Right-of-way at next intersection	Right-of-way at next intersection
Turn right ahead	Turn right ahead
Roundabout mandatory	Roundabout mandatory
Speed limit (30km/h)	Speed limit (30km/h)
Stop	Stop

My CNN was able to predict each of the new traffic signs correctly for 100% accuracy. This is similar to the test set accuracy of 0.917. The code for making predictions using my final model are in cells 12 - 14 in my Ipython notebook.

We can look at the softmax probabilities for each image to see how sure my model was on each image. This code can be found in cell 15 of my Ipython notebook. For all 5 images, the model was very sure about its answer. Here are the top 5 softmax probabilities for each image:

Prediction	Probability
Right-of-way at next intersection	1.00
Beware of ice/snow	2.18e-8
Pedestrians	5.14e-9
Speed limit (60km/h)	1.86e-11
Children crossing	1.03e-11

Prediction	Probability
Turn right ahead	1.00
Keep left	2.19e-9
General caution	1.76e-11
Go straight or left	9.48e-12
Bumpy road	7.88e-12

Prediction	Probability
Roundabout mandatory	0.997
Priority road	1.78e-3
Speed limit (120km/h)	2.97e-4
Right-of-way at next intersection	2.29e-4
Speed limit (100km/h)	1.11e-4

Prediction	Probability
Speed limit (30km/h)	1.00
Speed limit (80km/h)	4.44e-5
Speed limit (50km/h)	4.72e-7
Speed limit (20km/h)	1.99e-7
Speed limit (70km/h)	5.18e-8

Prediction	**Probability**
Stop	1.00
No entry	7.02e-7
Yield	2.65e-8
No vehicles	1.20e-8
Speed limit (30km/h)	8.96e-9