Traffic Sign Recognition Writeup

The writeup is briefly walking through the python notebook file "*Traffic_Sign_Classifier.ipynb*" of the project.

Data Set Summary & Exploration

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 use pickle to load the provided dataset. The datasets are already provided in three forms: train, valid and test. The benefit is that there is no need to split the training dataset in percentage such as 80 to 20 to have a training and validation datasets. After load the datasets, we know:

* The size of training set is 34799

* The size of the validation set is 4410

* The size of test set is 12630

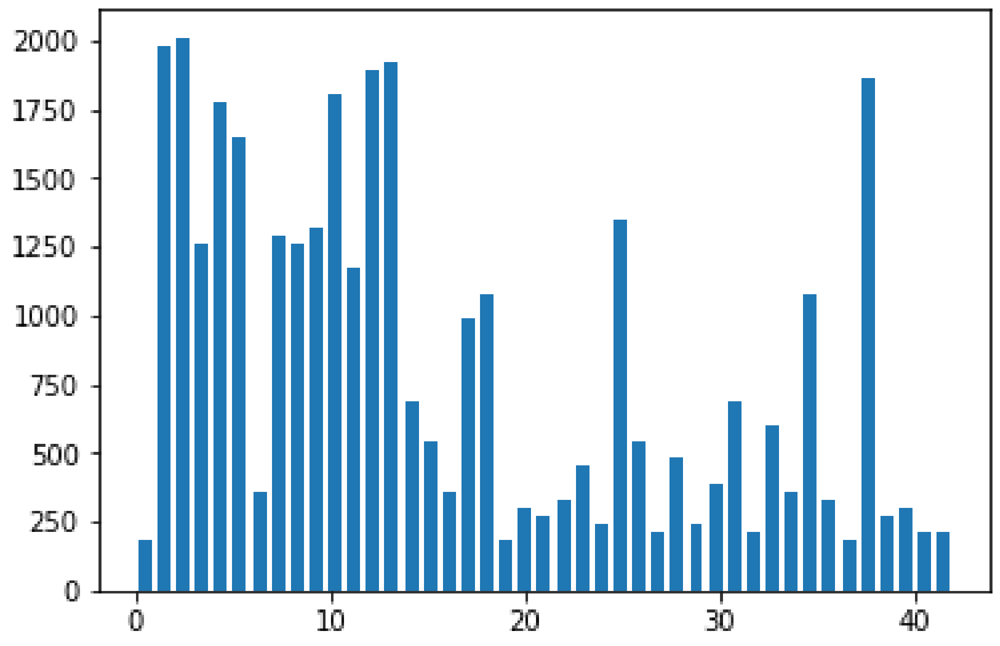
* The shape of a traffic sign image is (32, 32, 3)

* The number of unique classes/labels in the data set is 43

2. Include an exploratory visualization of the dataset.

Here is an exploratory visualization of the data set. Several samples of the traffic sign images:

Here is the bar chart of the traffic sign distribution: (43 classifications and the number of each class)



Design and Test a Model Architecture

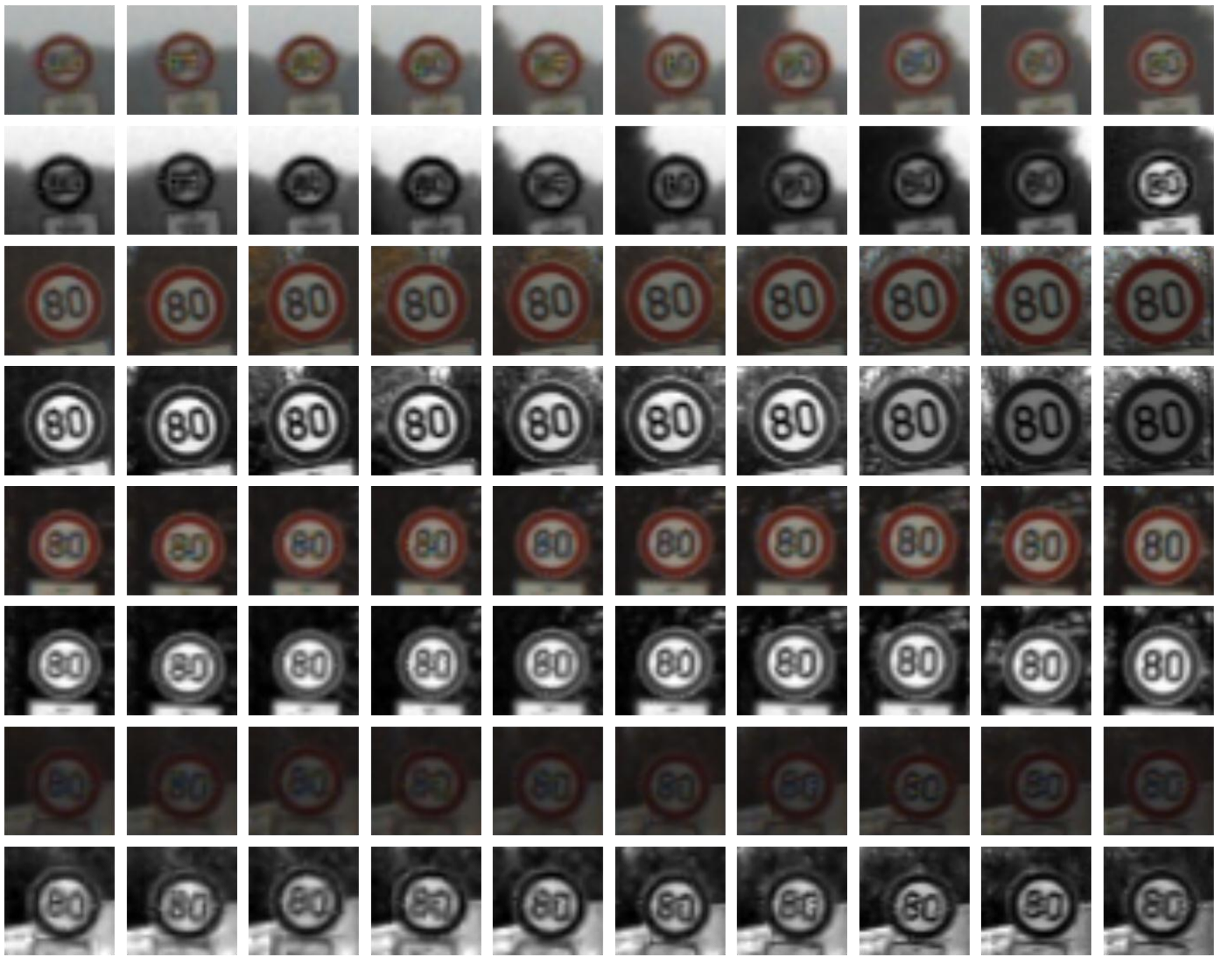
The neural network is designed from LeNet: https://github.com/udacity/CarND-LeNet-Lab/blob/master/LeNet-Lab-Solution.ipynb. The code is implemented in cell 8 and 9.

1. Describe how you preprocessed the image data. What techniques were chosen and why did you choose these techniques? Consider including images showing the output of each preprocessing technique. Pre-processing refers to techniques such as converting to grayscale, normalization, etc. (OPTIONAL: As described in the "Stand Out Suggestions" part of the rubric, if you generated additional data for training, describe why you decided to generate additional data, how you generated the data, and provide example images of the additional data. Then describe the characteristics of the augmented training set like number of images in the set, number of images for each class, etc.)

The model is from LeNet, because LeNet deals with single digits, so for traffic signs, it is necessary to convert the color from RGB to grayscale. This process will be done in the preprocess step.

As a first step, I decided to convert the images to grayscale as explained above. The preprocess function is defined in code cell 5.

Here is an example of a traffic sign image before and after grayscale.

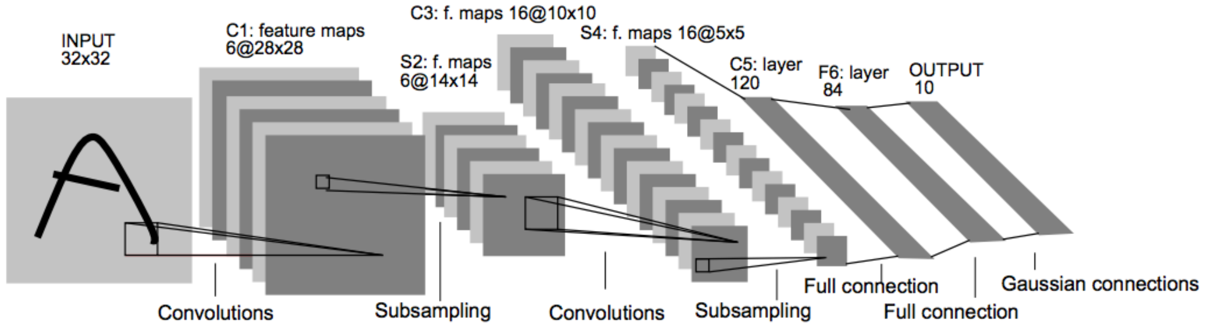


Then the next step, I normalized the image data the range of (-1, 1) is much better in computing than (0, 255).

There is another option to expand dataset by modifying the existing traffic signs, such as flip, reverse. But I prefer to test on the dataset first after the preprocess. If the result is not ideal, it may due to the lack of training data. Then the augmented image technic can be applied.

2. Describe what your final model architecture looks like including model type, layers, layer sizes, connectivity, etc.) Consider including a diagram and/or table describing the final model.

My final model is modified from LeNet as below, I added two dropout layers after the CNN and FN connect layers to reduce the computing complexity.



The model has the following layers:

1. 5x5 convolution (32x32x1 in, 28x28x6 out)
2. ReLU
3. 2x2 max pool (28x28x6 in, 14x14x6 out)
4. 5x5 convolution (14x14x6 in, 10x10x16 out)
5. ReLU
6. 2x2 max pool (10x10x16 in, 5x5x16 out)
7. Flatten layers (input 5x5x16, output 400)
8. Fully connected (input 400, output 120)
9. ReLu
10. Dropout
11. Fully connected (input 120, output 120)
12. ReLu
13. Dropout
14. Fully connected (input 84, output 43)

3. Describe how you trained your model. The discussion can include the type of optimizer, the batch size, number of epochs and any hyperparameters such as learning rate.

To train the model, I used the Adam optimizer with the parameters setting of:

- batch size: 100
- epochs: 60
- learning rate: 0.001
- mu: 0
- sigma: 0.1
- dropout keep probability: 0.5

4. Describe the approach taken for finding a solution and getting the validation set accuracy to be at least 0.93. Include in the discussion the results on the training, validation and test sets and where in the code these were calculated. Your approach may have been an iterative process, in which case, outline the steps you took to get to the final solution and why you chose those steps. Perhaps your solution involved an already well known implementation or architecture. In this case, discuss why you think the architecture is suitable for the current problem.

The provided input datasets are preprocessed at first, and then I shuffled the training and validation data to get a random input. I did not do further argumentation to expand the dataset, I would like to see the performance of only original data. Then I feed the training and validation data to the modified LeNet model to train the model and save the model for testing.

My final model results were:

* training/validation set accuracy of 0.966

* test set accuracy of 0.941

If an iterative approach was chosen:

* What was the first architecture that was tried and why was it chosen?

LeNet was chosen, because it is famous on digits classification and the traffic sign is similar to digits. LeNet takes image input that has only one color channel, so for traffic signs that has RGB color, it should convert to grayscale to directly use LeNet.

* What were some problems with the initial architecture?

The initial LeNet performance has the validation accuracy of 0.91 that is less than the target of 0.93.

* How was the architecture adjusted and why was it adjusted? Typical adjustments could include choosing a different model architecture, adding or taking away layers (pooling, dropout, convolution, etc), using an activation function or changing the activation function. One common justification for adjusting an architecture would be due to overfitting or underfitting. A high accuracy on the training set but low accuracy on the validation set indicates over fitting; a low accuracy on both sets indicates under fitting.

So the architecture was tuned a little bit by adding two dropout layers, each after the first and second fully connected layer.

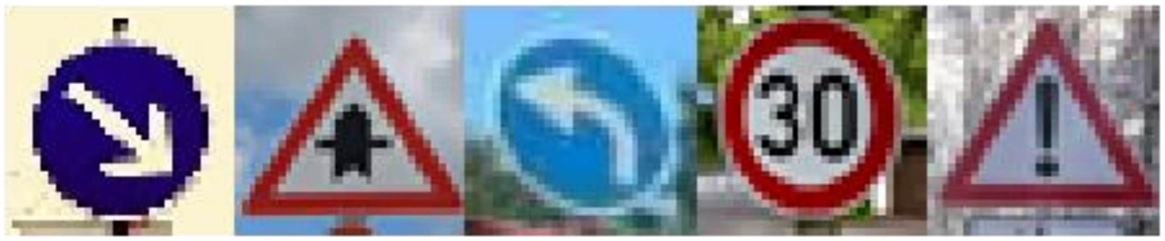
* Which parameters were tuned? How were they adjusted and why?

The mainly turned parameter the dropout probability (keep_prob), it was tuned from 0.2 to 0.5, and the compute speed was faster when keep_prob is 0.5, the result accuracy was also better.

Test a Model on New Images

1. Choose five German traffic signs found on the web and provide them in the report. For each image, discuss what quality or qualities might be difficult to classify.

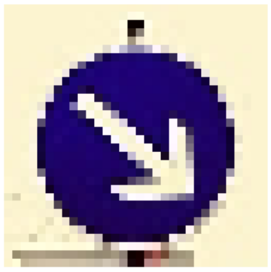
Here are five German traffic signs that I found on the web:

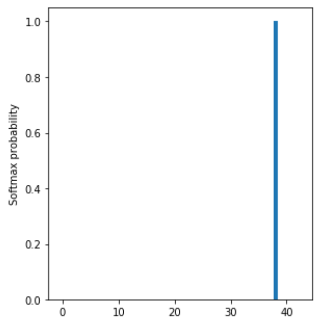


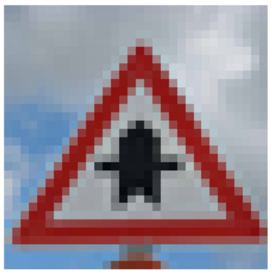
2. Discuss the model's predictions on these new traffic signs and compare the results to predicting on the test set. At a minimum, discuss what the predictions were, the accuracy on these new predictions, and compare the accuracy to the accuracy on the test set (OPTIONAL: Discuss the results in more detail as described in the "Stand Out Suggestions" part of the rubric).

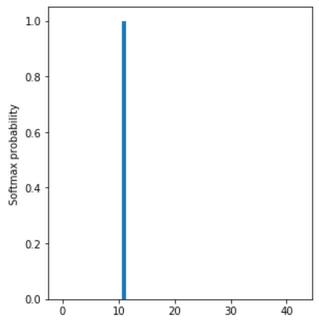
Here are the results of the prediction:

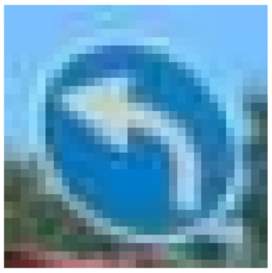
The result of the predication is 1, which means all five images are predicted correctly. Here is the bar chart of each image prediction:

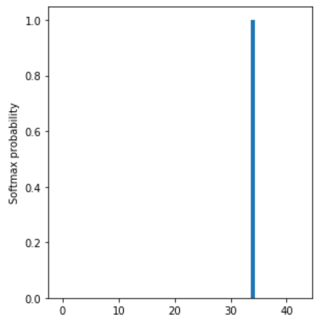


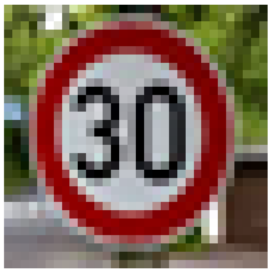

Softmax probability
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40

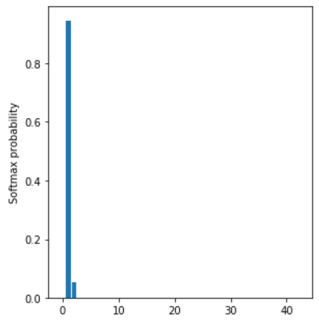


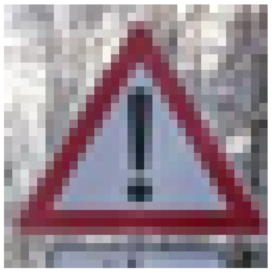

Softmax probability
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40

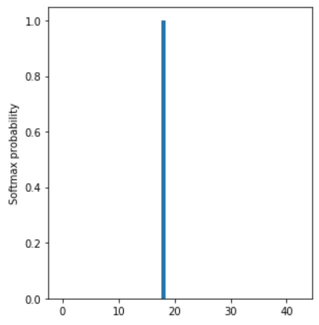



Softmax probability
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40




Softmax probability
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40




Softmax probability
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40

The model was able to correctly guess 5 of the 5 traffic signs, which gives an accuracy of 100%.

3. Describe how certain the model is when predicting on each of the five new images by looking at the softmax probabilities for each prediction. Provide the top 5 softmax probabilities for each image along with the sign type of each probability. (OPTIONAL: as described in the "Stand Out Suggestions" part of the rubric, visualizations can also be provided such as bar charts)

The code for making predictions on my final model is located in the cell 32 of the python notebook.