Project Report for Traffic Sign Recognition

Zhongzhou Yang

The goals / steps of this project are the following:

* 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

* Summarize the results with a written report

###Data Set Summary & Exploration

####1. Provide a basic summary of the data set and identify where in your code the summary was done. In the code, the analysis should be done using python, numpy methods rather than hardcoding results manually.

The code for this step is contained in the second code cell of the IPython notebook.

signs data set:

*** The size of training set is 34799**

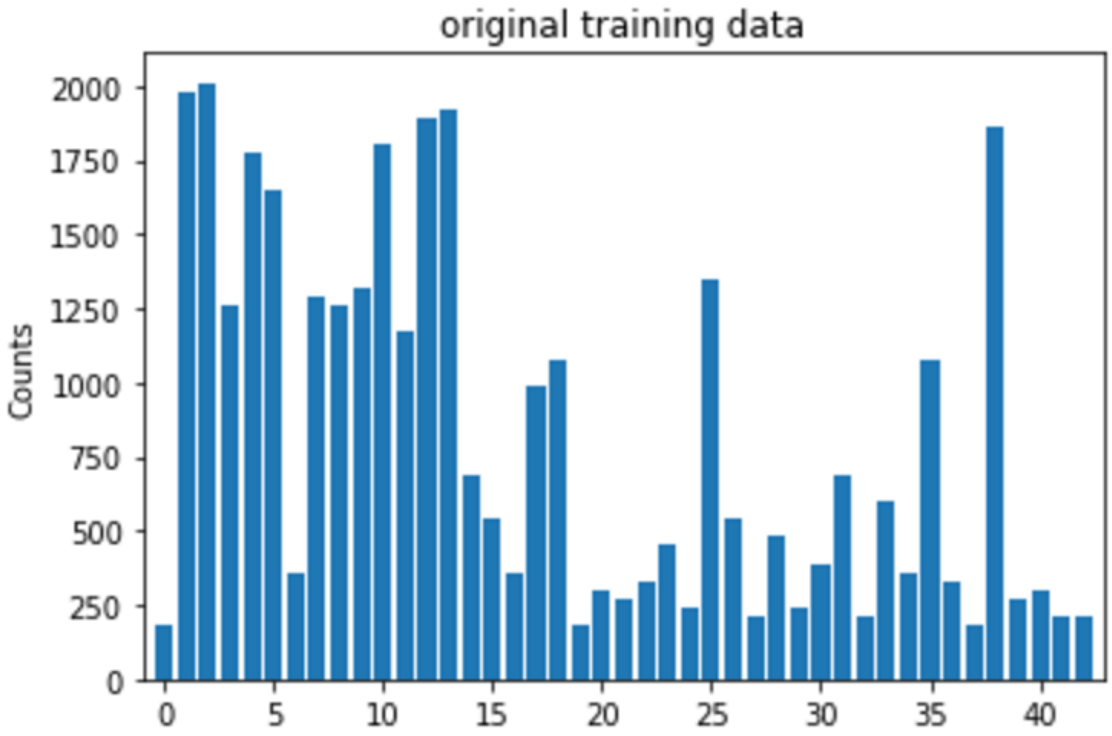
*** The size of test set is 12630**

*** 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 and identify where the code is in your code file. The code for this step is contained in the third code cell of the IPython notebook.

The bar chart in below shows the count of each sign from training data. This chart shows that the training data is unbalanced.



###Design and Test a Model Architecture

####1. Describe how, and identify where in your code, you preprocessed the image data. What tecniques 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. The code for this step is contained in the fourth code cell of the IPython notebook.

At first, convert colorful images to grayscale because

a. **Gray image is computationally cheaper**
b. **No big benefits to use colorful images according to the lectures and Yann LeCun's research findings**

The following figures show a colorful image converted into gray one.

Finally, image data are normalized because easier initialization and robust numerical stabilization. The data values are simply divided by 255, the maximum value in image data.

####2. Describe how, and identify where in your code, you set up training, validation and testing data. How much data was in each set? Explain what techniques were used to split the data into these sets. (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, identify where in your code, and provide example images of the additional data)

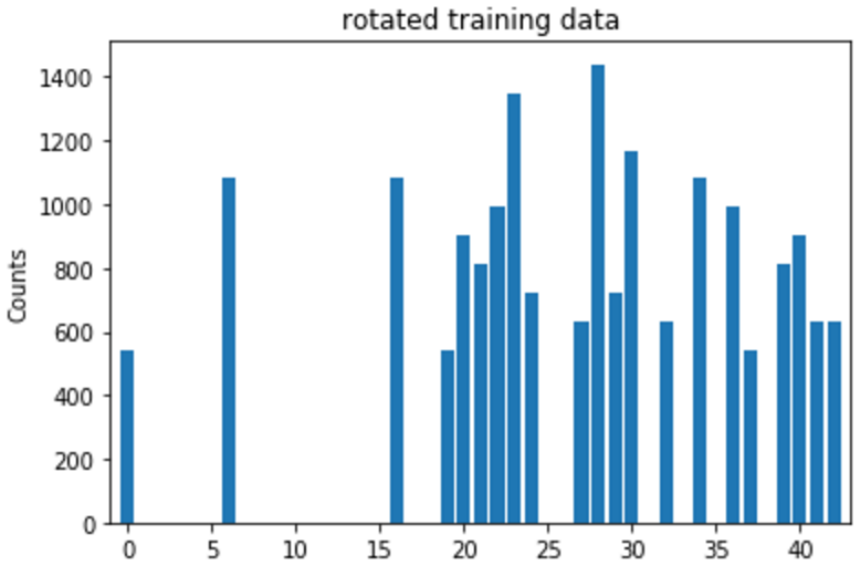
The fourth code cell of the IPython notebook contains the code for augmenting the data set. I decided to generate additional data because the data is unbalanced, which could lead serious bias after training.

To add more data to the data set, the following techniques were used:

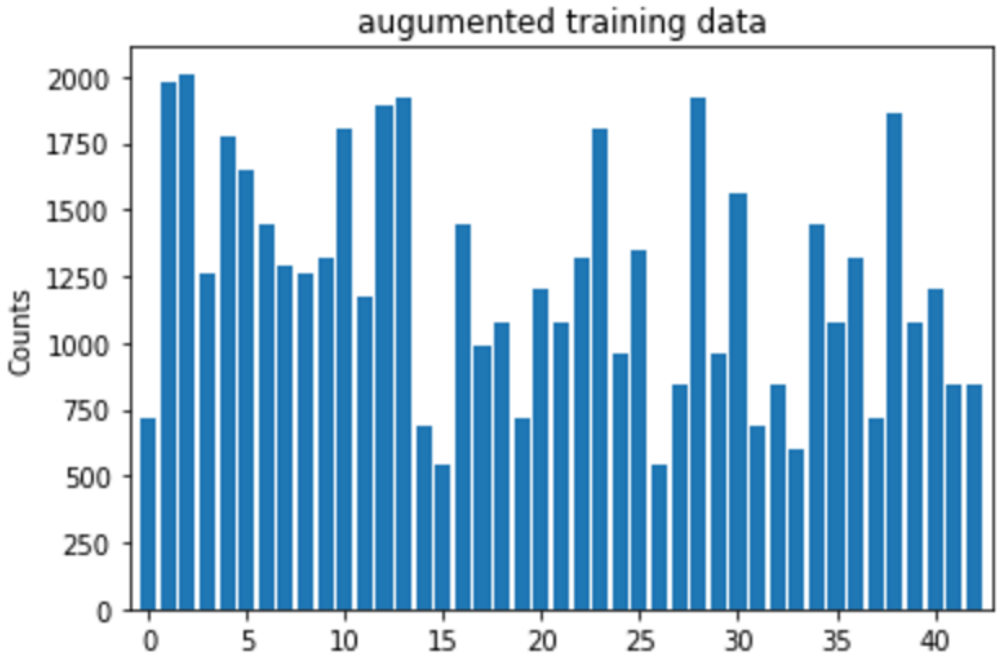
a) **Flipping the images**
b) **Rotating the images around 20 degrees**

The function of flipping images was borrowed from Alex Staravoitau's blog at
http://navoshta.com/traffic-signs-classification/

The images rotating function was borrowed from another blog. The reason, why the image is rotated around 20 degree, is that the sign number does not change if an image is rotated by a small angle. In order to generate a balanced training data set, image rotation were only applied to those images, which belong to the same traffic sign number and the total counting of these images is less than 500 for this kind of traffic sign. The following figure show that the distribution of rotated images across the traffic sign numbers.



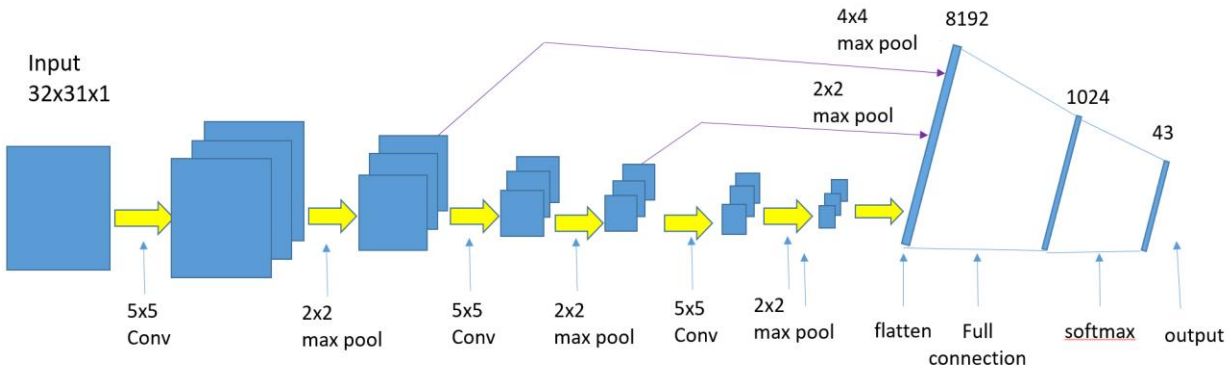
The following figure shows that image distributions with augmented rotated images. The image data set is more balanced than before.



####3. Describe, and identify where in your code, 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.

The code for my final model is located in the fifth cell of the ipython notebook.

My final model consisted of 3 convolutional layers and 1 fully-connected layer.



####4. Describe how, and identify where in your code, 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. The code for training the model is located in the sixth cell of the ipython notebook.

The Adam optimizer is used;

The batch size is 128;

The number of epochs is 35 due to using my local CPU for training; the epochs number could be set to much larger once setting of the AWS instances are finished;

The learning rate is set to 0.001 for the first 20 epochs; the learning rate is set to 0.0001 after that to reach higher accuracy;

####5. Describe the approach taken for finding a solution. 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 code for calculating the accuracy of the model is located in the 6th cell of the Ipython notebook.

My final model results were:

*** training set accuracy of 0.993**

*** validation set accuracy of 0.957**

*** test set accuracy of 0.937**

If an iterative approach was chosen:

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

The LeNet architecture was tried first with final output number changed to 43. This is a reasonable and easy starting point.

* What were some problems with the initial architecture?

The highest accuracy is 0.92; overfitting always exists.

* 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 over fitting or under fitting. 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.

In order to overcome overfitting, tricks including dropout and L2 regularization are used;

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

Learning rate was adjusted lower to achieve high accuracy;

* What are some of the important design choices and why were they chosen? For example, why might a convolution layer work well with this problem? How might a dropout layer help with creating a successful model?

Finally, multi-scale convolutional network is chosen because it is very difficult to reach accuracy above 0.93 by use of strict feed-forward CNNs;

If a well known architecture was chosen:

* What architecture was chosen?

The architecture of the following paper is chosen.

Pierre Sermanet and Yann LeCun, ”Traffic Sign Recognition with Multi-Scale Convolutional Networks”, 2011.

* Why did you believe it would be relevant to the traffic sign application?

This paper showed improved capability by use of multi-scale convolutional networks.

* How does the final model's accuracy on the training, validation and test set provide evidence that the model is working well?

By use of multi-scale convolutional networks, the validation accuracy could easily reach above 0.93 within 20 epochs.

###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.

I randomly chose images from number 28 to 32.

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

The first image might be difficult to classify because it is blurred.

####2. Discuss the model's predictions on these new traffic signs and compare the results to predicting on the test set. Identify where in your code predictions were made. 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).

The code for making predictions on my final model is located in the 9th cell of the Ipython notebook.

Here are the results of the prediction:

Image	Prediction
No passing	No passing
Right-of-way at the next intersection	Right-of-way at the next intersection
Speed limit (80km/h)	Speed limit (80km/h)
No entry	No entry
Turn left ahead	Turn left ahead

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 and identify where in your code softmax probabilities were outputted. 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 10th cell of the Ipython notebook.

Probability for 4 of 5 images are nearly 1.0;

Only the 3rd one, the probability is 9.99983907e-01; See details in below:

Probability	Prediction
1.0	No passing
1.0	Right-of-way at the next intersection
9.99983907e-01	Speed limit (80km/h)
1.0	No entry
1.0	Turn left ahead