Traffic Sign Classifier Project

DEEP LEARNING

Kirsten Bailey | Self-Driving Car Engineer Nanodegree | 11/2/2017

Traffic Sign Recognition

Writeup Template

You can use this file as a template for your writeup if you want to submit it as a markdown file, but feel free to use some other method and submit a pdf if you prefer.

Build a Traffic Sign Recognition Project

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

RUBRIC POINTS

Here I will consider the rubric points individually and describe how I addressed each point in my implementation.

Writeup / README

1. Provide a Writeup / README that includes all the rubric points and how you addressed each one. You can submit your writeup as markdown or pdf. You can use this template as a guide for writing the report. The submission includes the project code.

You're reading it! Here is a link to my project code

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 used the pandas library to calculate summary statistics of the traffic signs data set:

- The size of training set was 34,799
- The size of the validation set was 4,410
- The size of test set was 12,630

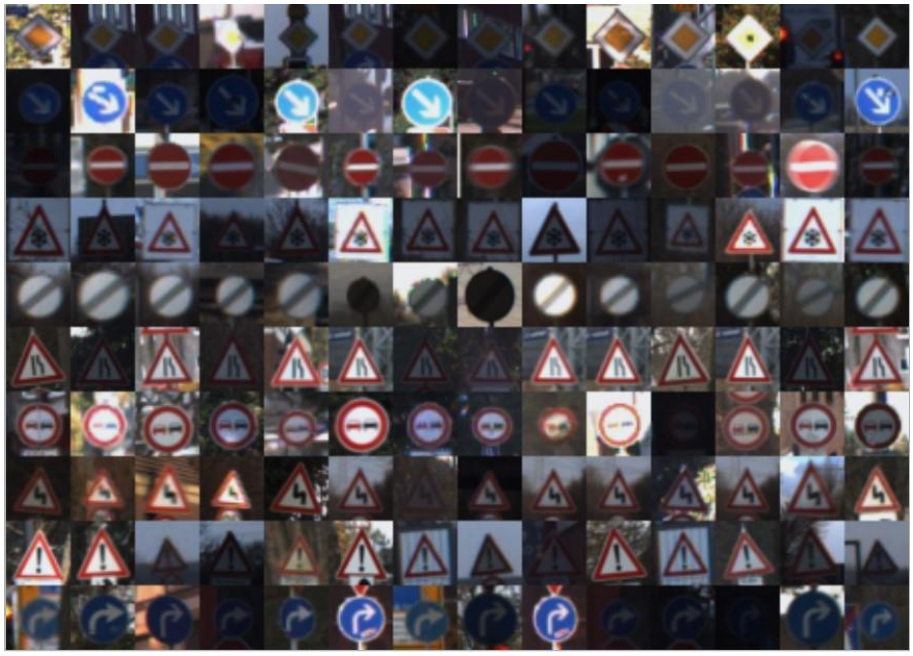
- The shape of a traffic sign image was (32, 32, 3)
- The number of unique classes/labels in the data set was 43 (e.g. Speed Limit 20km/h, Wild Animals Ahead, No Passing, etc.)

I used Python 3.5 with Tensorflow to write the code.

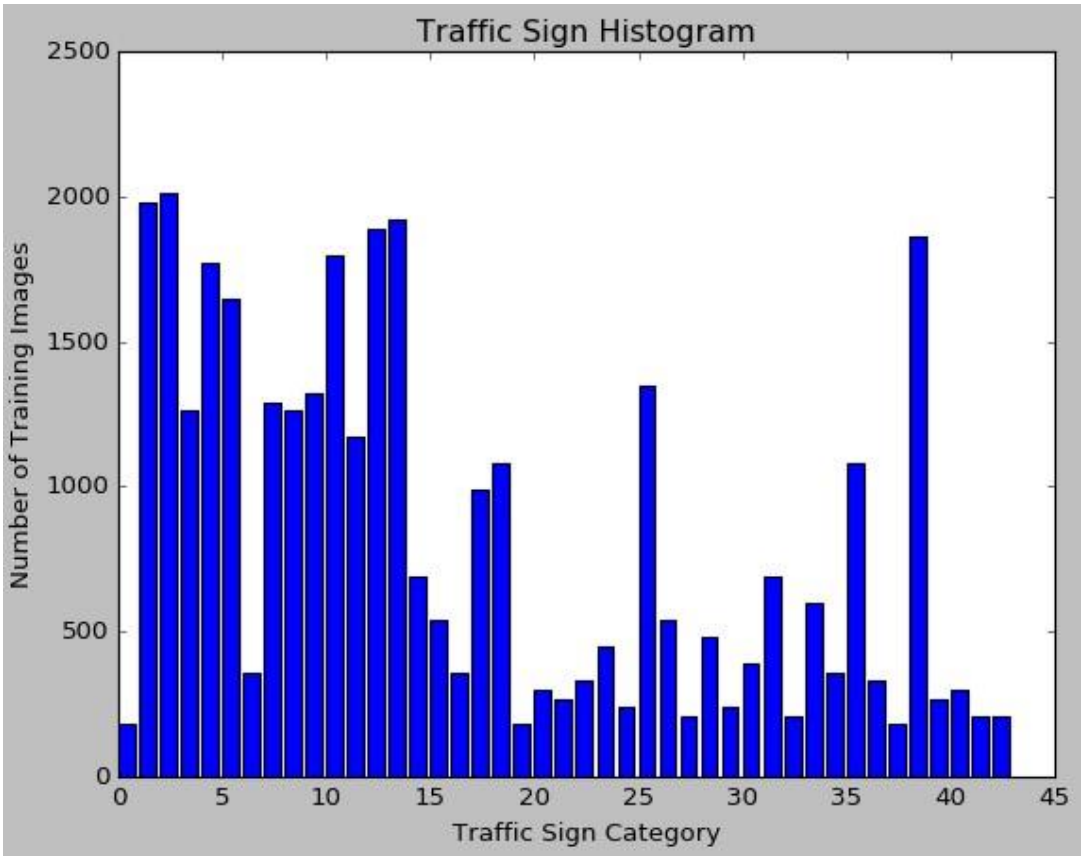
2. Include an exploratory visualization of the dataset.

Here is an exploratory visualization of the data set. It contains sample color images along with their individual description of the traffic sign image from the dataset.

I also included random color images in a grid pattern. Some of the images appear to be very dark while some are very light so we will look to improve the contrast later.



Additionally, I included a histogram bar chart showing how the training dataset is unevenly distributed over each category of signage. The histogram shows that some classes have fewer than 200 images while other classes contain over 1800 images. This discrepancy indicates that our model can be biased towards highly-represented classes, particularly when the model is uncertain in its predictions. We will use data augmentation later to mitigate this class-size discrepancy.

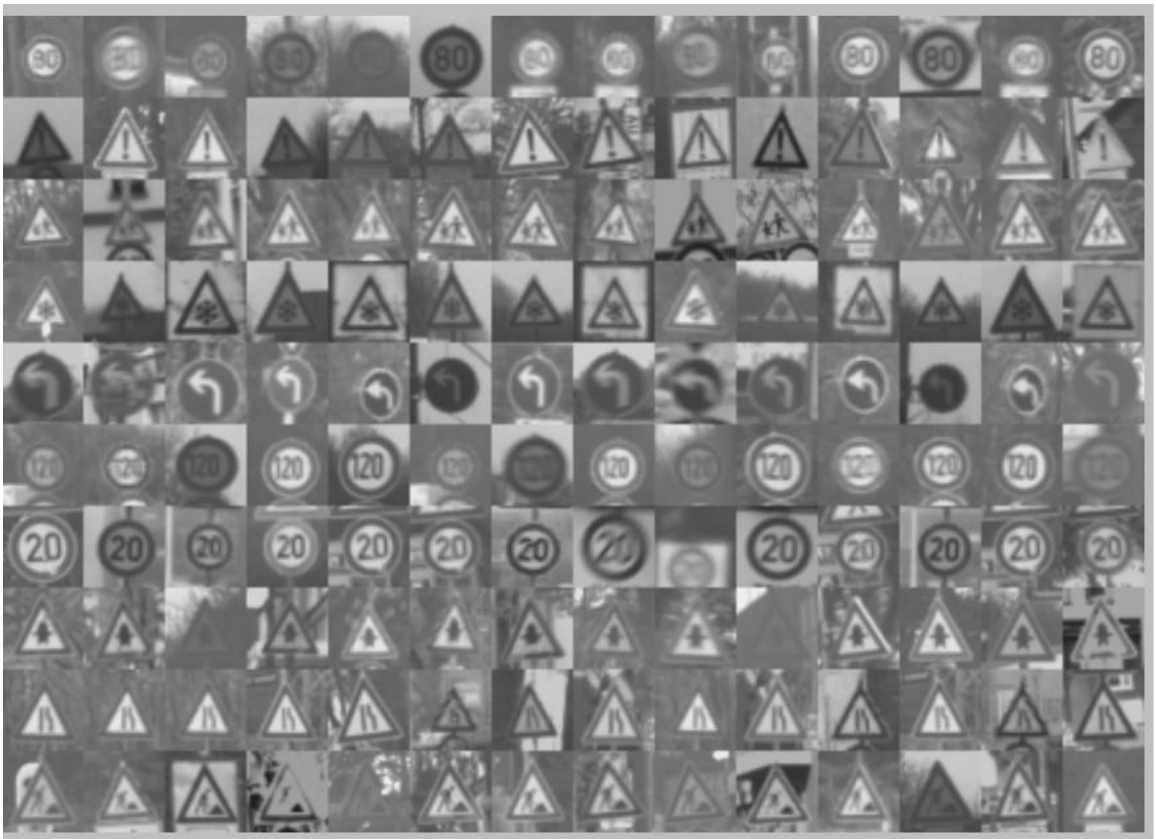


Design and Test a Model Architecture

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.)

As a first step, I decided to convert the three-channel color images to grayscale because just as in project 1 – Lane Line Detection Project – we are interested in increasing the image's contrast while at the same time, reducing the noise in an image prior to running

more powerful algorithms that will determine the correct shapes and lines that identify the appropriate traffic sign.



Random Images After Pre-processing

After converting to grayscale, the accuracy of the model did not significantly change, however, it did make it easier to complete normalization.

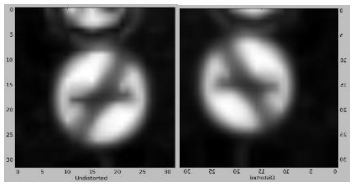
I then saturated the intensity values at 1 and 99 percentiles, set the min/max normalization to the range of [0,1] and subtracted its mean from the image, making the values centered around zero.

With a simple min/max normalization, there was an approximately 1% improvement in validation accuracies than without it. This percentile-based methodology gave an approximately 1% improvement over straight-forward min/max normalization.

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

As a last step, I normalized the image dataset by subtracting each image by the dataset's mean and then divided by its standard deviation. This normalization process helps the model treat images in a more uniform manner.

Here is an example of an original image and an augmented image:



The difference between the original data set and the augmented dataset is the following:

a) I added 2 distorted versions of each training image to the training set by adding a random rotation of (-6, 6 degrees) and a random uniform translation of (-0.1, .1) pixels both horizontally and vertically. I also added a random scale of the range (0.9, 1.1) to the model.
b) After distortion, some empty pixels were filled with "edge" padding.

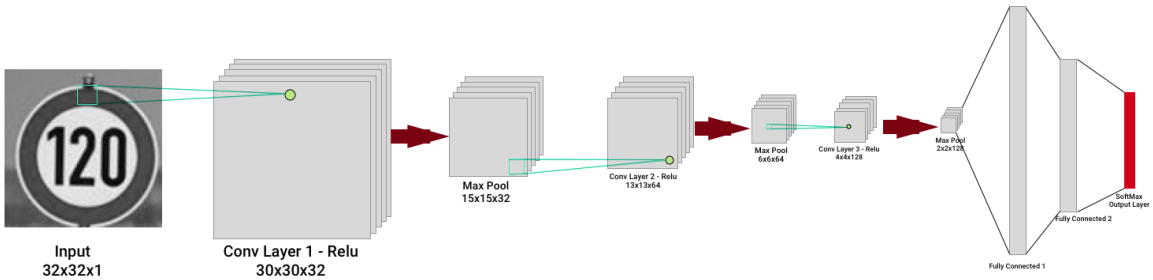
2. Describe what your final model architecture looks like including model type, layers, layer sizes, connectivity, etc.) Consider including a diagram and/or table describing the final model.

My final model consisted of the following layers:

Layer	Description
1	Convolutional (5 x 5 x 1 x 32) Input = 32 x 32 x 1. Output = 28 x 28 x 32. Relu activation Pooling. Input = 28 x 28 x 32. Output = 14 x 14 x 32.
2	Convolutional (5 x 5 x 32 x64) Input = 14 x 14 x 32. Output = 10 x 10 x 64. Relu activation Pooling. Input = 10 x 10 x 64. Output = 5 x 5 x 64
3	Flatten the image. Output = 5 x 5 x 64. Output = 1600. Dropout = 0.7 Fully Connected. Input = 1875. Output = 120.

Layer	Description
	Relu activation
	Dropout = 0.7
	Fully Connected. Input – 120. Output = 84
4	Relu activation
	Dropout = 0.7
5	Fully Connected. Input = 84. Output = 43

The architecture used is a deep convolutional neural network that was inspired by two existing achitectures: LeNet and Ciresan, although while Ciresan's architecture properties contain a multi-column network, mine is much simpler (and probably less accurate) however it makes training and prediction much faster.



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 a TensorFlow Adam Optimizer, which is a more sophisticated Gradient Descent Optimizer on a batch size of 128. I ran 25 epochs at a learn rate of 0.001.

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.

I began by creating an architecture that could overfit the training data – a few epochs had a 1.000 training accuracy however, the validation accuracy was much lower. I then changed the parameters until I was able to nearly eliminate the overfitting. I added L2 regularization for the weights and dropout operations between the fully connected layers, however, the accuracy level only decreased by a small amount.

My final model results were:

- training set accuracy of 0.999
- validation set accuracy of 0.988
- test set accuracy of 0.971

If an iterative approach was chosen:

- What was the first architecture that was tried and why was it chosen?
- What were some problems with the initial architecture?
- 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.
- Which parameters were tuned? How were they adjusted and why?
- 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?

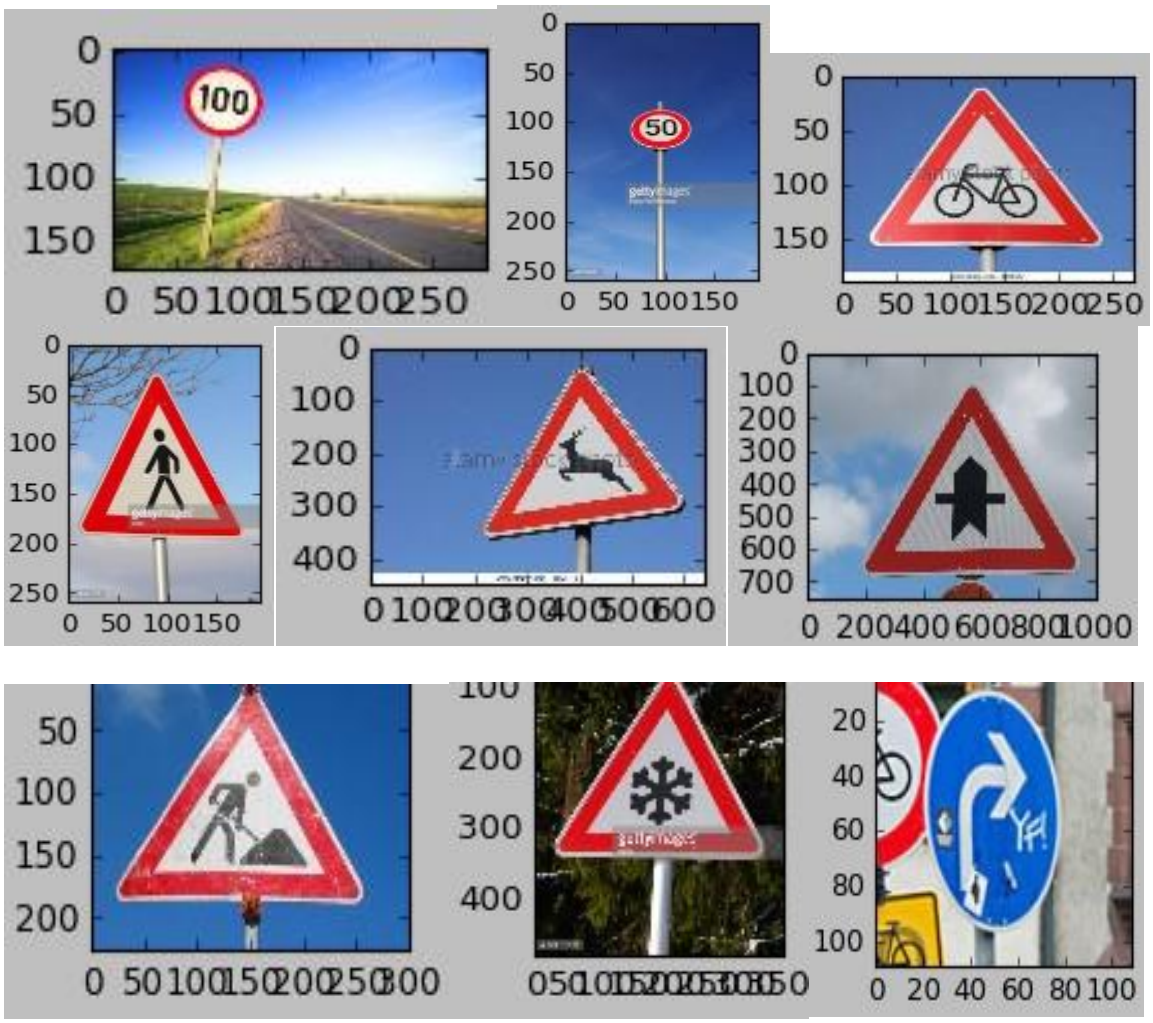
If a well-known architecture was chosen:

- What architecture was chosen?
- Why did you believe it would be relevant to the traffic sign application?
- How does the final model's accuracy on the training, validation and test set provide evidence that the model is working well?

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 nine German traffic signs that I found on the web:



Beginning from the top left and moving to the right, the first image (100 km/h) might be difficult to classify because of the sign's angle.

The second image might be difficult to classify because of the sign's angle and the fact that there is copy printed on top of the image, albeit not directly on the sign image.

The third, fourth and fifth and eighth images of the bicycle, pedestrian, deer ahead, and ice ahead, respectively, have copy printed directly over the top of the image and this could cause some confusion for the model.

With the exception of image nine, the other images should be straight-forward for the model to read accurately. Image nine not only contains remnants of other signage in a cluster, but the right turn ahead sign contains some graffiti, which could cause some confusion for the model.

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:

Image	Prediction
Speed Limit (100 km/h)	No Passing (Incorrect)
Speed Limit (50 km/h)	Vehicles Over 3.5 Metric Tons Prohibited (Incorrect)
Bicycle Ahead	Bicycle Ahead
Pedestrian Ahead	Pedestrian Ahead
Wild Animals Ahead	Wild Animals Ahead
Right of Way at Next Intersection	Right of Way at Next Intersection
Right-of-Way Next Crossroads	Yield (Incorrect)
Road Work Ahead	Road Work Ahead
Ice Ahead	Ice Ahead

The model was able to correctly predict 6 of the 9 traffic signs, which gives an accuracy of 77.78%. This is much lower than the test accuracy of 97.1%, however, this is a nine-image dataset and is too small to represent its ability. If I were to run this code again on new images from the web, I would look for signs with more anomalies such as dark signs, signs obscured by branches, more signs with graffiti, etc., which more accurately represents images that are contained in the training dataset.

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 12th cell of the Ipython notebook.

For the first image, the model is relatively sure that this is a “No Passing” Sign (probability of 0.557). The top five Softmax probabilities were

Probability	Prediction
0.5557	~~No Passing~~ (Speed Limit 100km/h)
0.2213	Speed Limit 120 km/h
0.1151	No Vehicles
0.0465	End of No Passing
0.0148	End of All Speed & Passing Limits

For the second image, the model was 64.58% sure that “Vehicles Over 3.5 Metric Tons Prohibited,” however, the traffic sign was a speed limit sign of 50 km/h. That speed limit sign was not even in the top-five Softmax probabilities.

Probability	Prediction
0.6458	~~Vehicles Over 3.5 Metric Tons Prohibited~~ (Speed Limit 50 km/h)
0.1173	No Entry
0.0878	Speed Limit (80km/h)
0.0374	Speed Limit (100km/h)
0.0146	Dangerous Curve to the Right

The third image was of “Bicycle Crossing” was correctly predicted at 99.96% accuracy.

Probability	Prediction
0.9996	Bicycle Crossing
0.0003	Children Crossing
0.0000	Road Narrows to the Right
0.0000	Slippery Road
0.0000	Bumpy Road

Image four was a sign image of Pedestrians which the model identified with 100% accuracy.

Probability	Prediction
1.0000	Pedestrians
0.0000	Right-of-Way at Next Intersection
0.0000	Road Narrows on the Right
0.0000	Children Crossing
0.0000	General Caution

Image five was “Wild Animals Crossing.” The model predicted the correct sign with 99.93% accuracy.

Probability	Prediction
0.9993	Wild Animals Crossing

Probability	Prediction
0.0004	Double Curve
0.0003	Dangerous Curve to the Left
0.0000	Slippery Road
0.0000	General Caution

Image six was correctly identified with 100% accuracy to be “Right-of-Way at the Next Intersection.” The error here was a human one: I inputted “13” instead of “11” causing the model to call this an error. It is not!

Probability	Prediction
1.0000	Right-of-Way at the Next Intersection
0.0000	Double Curve
0.0000	Beware of Ice/Snow
0.0000	Pedestrians
0.0000	Roundabout Mandatory

With image seven, “Road Work Ahead,” the model correctly predicted the image with 99.82% accuracy.

Probability	Prediction
0.9982	Road Work
0.0012	Road Narrows on the Right

Probability	Prediction
0.0006	Bicycles Crossing
0.0000	Bumpy Road
0.0000	Beware of Ice/Snow

For the eighth image, the mode correctly identified with 97.81% accuracy the “Beware of Ice/Snow” traffic sign image.

Probability	Prediction
0.9781	Beware of Ice/Snow
0.0149	Bicycles Crossing
0.0052	Road Wok
0.0006	Priority Road
0.0006	Slippery Road

For the ninth and final image, the model predicted “Turn Right Ahead” traffic sign with 99.96% accuracy.

Probability	Prediction
0.9996	Turn Right Ahead
0.0001	Priority Road
0.0001	Speed Limit (60km/h)
0.0001	Stop

Probability	Prediction
0.0001	Ahead Only

(Optional) Visualizing the Neural Network (See Step 4 of the Ipython notebook for more details)

1. Discuss the visual output of your trained network's feature maps. What characteristics did the neural network use to make classifications?

In visualizing the activations of the first convolutional layer, I recognized that different feature maps look for different edges, for example the 9th feature map is activated by diagonal edges while the 14th feature map is activated by horizontal edges.

Reference

[1] Lecun (1998): Gradient-Based Learning Applied to Document Recognition

[2] Sermanet (2011): Traffic Sign Recognition with Multi-Scale Convolutional Networks

[3] Ciresan (2012): Multi-Column Deep Neural Network for Traffic Sign Classification