ameter / CarND-TrafficSigns-P2

CarND Project 2

Edit

Manage topics

20 commits | 1 branch | 0 releases | 1 contributor

Branch: master | New pull request | Create new file | Upload files | Find File | Clone or download

ameter cleaned up repo — Latest commit 187153f on Nov 28, 2017

test_images	added and processed new test images from the web.	a year ago
writeup_images	tweaked notebook and added class_counts image.	a year ago
README.md	Update README.md	a year ago
Traffic_Sign_Classifier.html	added labels output. cleaned up softmax output formatting.	a year ago
Traffic_Sign_Classifier.ipynb	added labels output. cleaned up softmax output formatting.	a year ago
checkpoint	added trained model data to repo	a year ago
lenet.data-00000-of-00001	added trained model data to repo	a year ago
lenet.index	added trained model data to repo	a year ago
lenet.meta	added trained model data to repo	a year ago
signnames.csv	Added signnames file to repo	a year ago

README.md

Traffic Sign Recognition

Writeup

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! and here is a link to my project code

Data Set Summary & Exploration

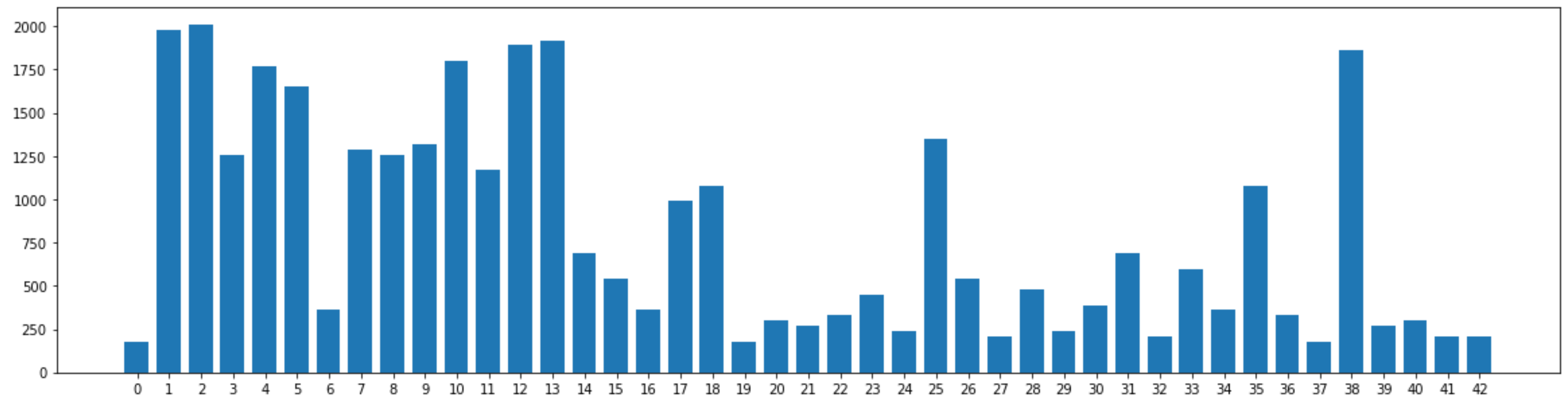
1. Provide a basic summary of the data set. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.

I used raw python and numpy to calculate summary statistics of the traffic signs data set:

- 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. It is a bar chart showing how the counts of each sign class in the training set.



I also displayed some random images and labels from the training set to make sure they matched up correctly and validate my input processing.

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.)

I normalized the image data to make it zero mean and low variance. I actaully had a lot of trouble with this step because after normalization, accuracy went way down. After many hours trying to figure out why, I finally realized it was a Pythonism. I was using (pixel - 128) / 128 for normalizations, so it was truncating things to ints and causing weird results. I changed it (pixel - 128.0) / 128.0 and accuracy improved dramatically.

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
Input	32x32x3 RGB image
1. Convolution	Input = 32x32x3. Output = 28x28x6.
1. RELU	
1. Max pooling	Input = 28x28x6. Output = 14x14x6.
1. Dropout	
2. Convolution	Input = 14x14x6. Output = 10x10x16.
2. RELU	
2. Max pooling	Input = 10x10x16. Output = 5x5x16.
2. Dropout	
3. Fully connected	Input = 400. Output = 120.
3. RELU	
3. Dropout	
4. Fully connected	Input = 120. Output = 84.
4. RELU	
4. Dropout	
5. Fully connected	Input = 84. Output = 43.
Output	Logits for 43 classes

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 an Amazon AWS g2.2xlarge GPU instance based on the udacity-carnd AMI. I experimented with a number of batch sizes and epochs and tried a few different learning rates. Most of my training runs were 50 or 100 epochs. However, on my final run, after I had everything else optimized, I ran for 500 epochs and achieved slightly higher accuracy with my final model. I also tried a few different batch sizes and learning rates and settled on a batch size of 128 and a learning 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.

My final model results were:

- validation set accuracy of 0.962
- test set accuracy of 0.952

I began with the LeNet-5 architecture from the CNN lab. I felt it was a reasonable place to start because it worked well with other image classification use cases. When I first implemented it and trained it with the street sign data set, it achieved approximately 95% accuracy on the valication set out of the box. I thought this was great and would not require any further optimization to meet the project goal of greater than 93% accuracy. Thinking I was done, I ran it on the test set and only got 86% accuracy. It was clearly overfitting and needed more work, so I regretted running the test set, oops!

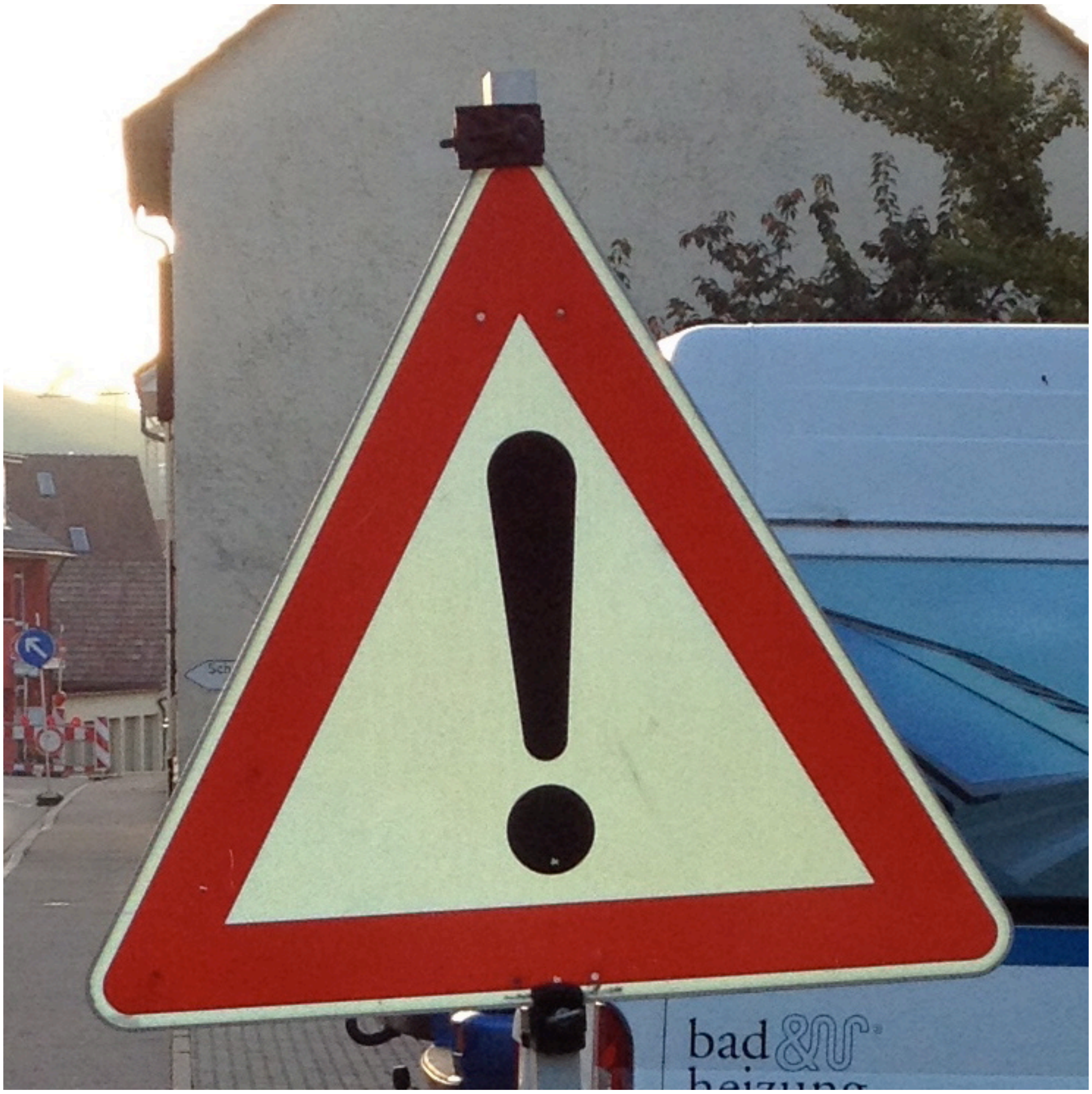
The major improvment I made to the LeNet-5 architecture was adding dropout to each layer. This greatly helped with the overfitting problem, but now my validation accuracy was lower. I then normalized my input data, and after correcting the normalization to (pixel - 128.0) / 128.0 as described above, my accuracy improved dramatically.

Given that the final scores were close between validation and test accuracy (0.962 and 0.952, respectively), it appears that adding dropout did in fact resolve the overfitting issue.

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 six German traffic signs that I found on the web:



All of these images seemed like they should be relatively easy to classify because they were cropped to only include the sign and they were relatively clear and squarly oriented.

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
General caution	General caution
No entry	No entry
No passing	No passing

Priority road	Priority road
Stop	Stop
End of all speed and passing limits	End of all speed and passing limits

The model was able to correctly guess 6 of the 6 traffic signs, which gives an accuracy of 100%. This compares favorably to the accuracy on the test set of 0.952.

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 11th cell of the Ipython notebook.

For the first image, the model is highly confident that this was a general caution sign (probability of 1.0), and the image does contain a general caution sign. The top five soft max probabilities were

Probability	Prediction
1.0	General caution
3.54746e-09	Traffic signals
2.8323e-13	Pedestrians
1.36002e-18	Road narrows on the right
5.36388e-23	Right-of-way at the next intersection

For the second image, the model is highly confident that this was a No entry sign (probability of 1.0), and the image does contain a No entry sign. The top five soft max probabilities were

Probability	Prediction
1.0	No entry
4.03275e-22	Stop
8.01291e-23	No passing
5.98542e-28	Speed limit (20km/h)
2.28292e-32	No passing for vehicles over 3.5 metric tons

For the third image, the model is highly confident that this was a No passing sign (probability of 1.0), and the image does contain a No passing sign. The top five soft max probabilities were

Probability	Prediction
1.0	No passing
3.24911e-16	No vehicles
4.38651e-17	Vehicles over 3.5 metric tons prohibited
3.12406e-17	No passing for vehicles over 3.5 metric tons
1.84159e-20	Speed limit (60km/h)

For the fourth image, the model is highly confident that this was a Priority road sign (probability of 1.0), and the image does contain a Priority road sign. The top five soft max probabilities were

Probability	Prediction
1.0	Priority road
1.52736e-15	Stop
1.11633e-17	Keep right
9.0961e-18	Speed limit (30km/h)
1.32395e-18	Speed limit (80km/h)

For the fifth image, the model is highly confident that this was a Stop sign (probability of 1.0), and the image does contain a Stop sign. The top five soft max probabilities were

Probability	Prediction
1.0	Stop
3.60673e-10	Speed limit (30km/h)
3.80721e-13	Yield
1.64907e-13	Speed limit (20km/h)
1.42384e-13	Speed limit (50km/h)

For the sixth image, the model is highly confident that this was a End of all speed and passing limits sign (probability of 0.976576), and the image does contain a End of all speed and passing limits sign. The top five soft max probabilities were

Probability	Prediction
0.976576	End of all speed and passing limits
0.0232536	End of no passing
9.91662e-05	Go straight or right
4.7699e-05	End of speed limit (80km/h)
1.16619e-05	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?

Due to time, I did not attempt the optional portion of the project. I plan to come back to it after the term over the holidays because I'm very interested in it.