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! Project uploaded via portal.

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.

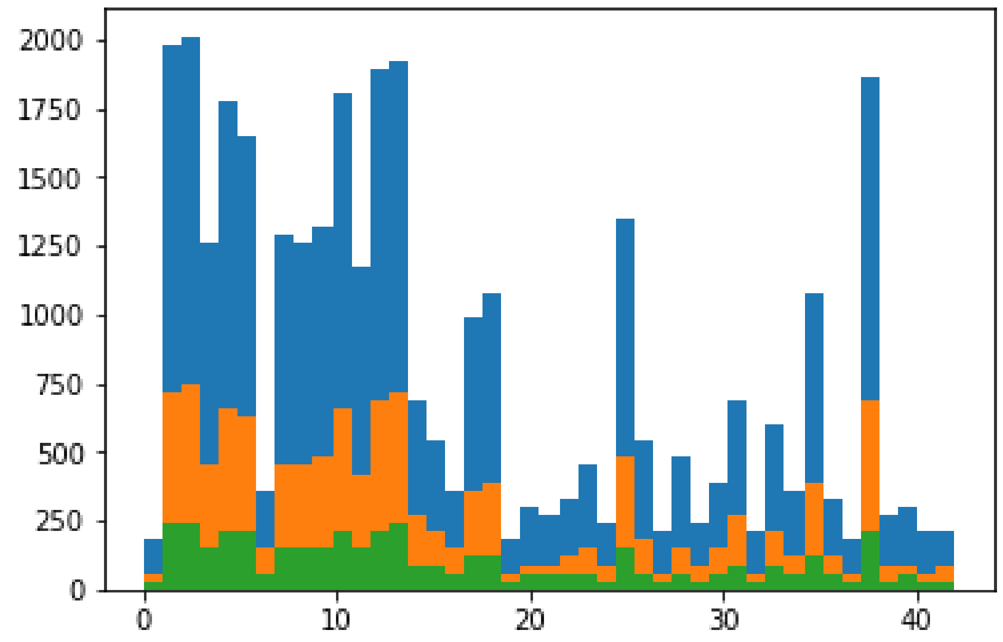
I used the pandas library to calculate summary statistics of the traffic signs data set:

- Number of training examples = 34799
- Number of testing examples = 12630

- Image data shape = (32, 32)

2. Include an exploratory visualization of the dataset.

Here is an exploratory visualization of the data set. First I checked the random images to explore how the images look. Then curious about the distribution of the data in various classes in train, validation and test set.



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 wanted to perform data augmentation and yield more samples for my training set. Hence used the library “Augmentor” which makes it super easy to create images. The actual training set was used a starting point, random images were chosen and “rotation” was applied. This will help us train the model with more diverse images and won’t let the model overfit. Then I decided to convert the images to grayscale because it’s a common technique used to reduce input dimensions. Also I found that each pixel value “averages out” when you convert to grayscale. This will help in the next step which is normalization where in each pixel values is normalized by the formula

(gray_image - gray_image.min()) / (gray_image.max () - gray_image.min ())

This will ensure the values are scaled down between 0 and 1 which help in model speed, accuracy and convergence.

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.

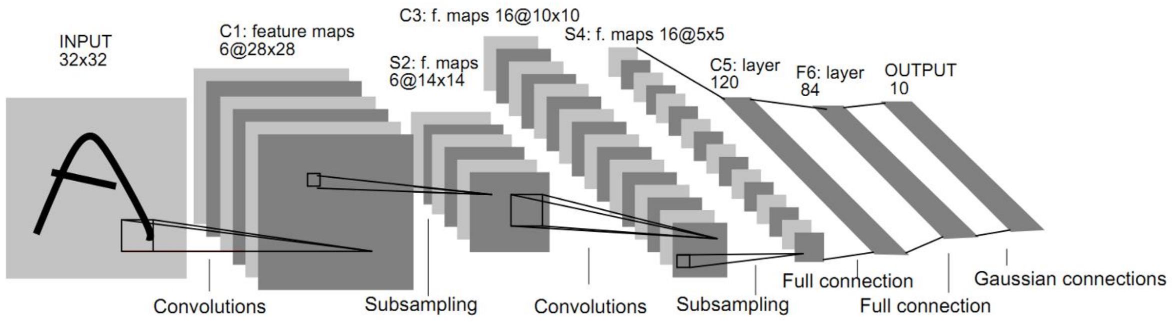


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

My final model consisted of the following layers: I used a classic LeNet Architecture which accepts 32x32x1 as input and outputs 43 logits.

Layer	Description
Input	32x32x1 RGB image
Convolution 5x5	1x1 stride, VALID padding, outputs 32x32x6
RELU	
Max pooling	2x2 stride, outputs 14x14x6

Layer	Description
Convolution 5x5	1x1 stride, VALID padding, outputs 10x10x16
RELU	
Max pooling	2x2 stride, outputs 5x5x16
flatten	Input = 5x5x16. Output = 400
Fully Connected	Input = 400. Output = 120.
RELU	
Fully Connected	Input = 120. Output = 84.
RELU	
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 a batch size of 128. I experimented with epochs and finally arrived at 50. Learning rate was set to 0.001. This helped achieve and accuracy rate of 95%.

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:

- Training loss of almost **0%**
- validation set accuracy of **95.7%**
- Test set accuracy of **93.2%**

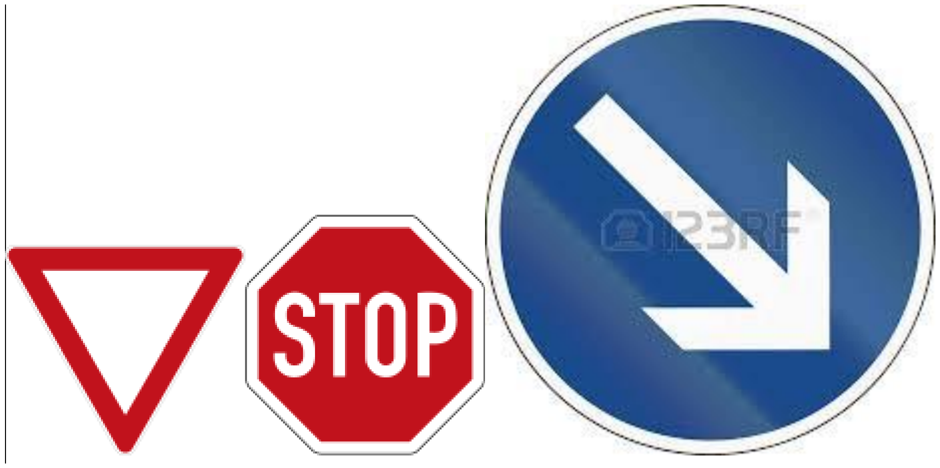
A well-known architecture was chosen:

- LeNet Architecture was chosen as a starting point. It worked on MNIST dataset and the traffic sign data had a very similar dataset i.e. multi-classification problem on a set of images with similar size and resolution.
- How does the final model's accuracy on the training, validation and test set provide evidence that the model is working well?

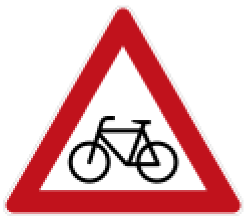
Test a Model on New Images

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.

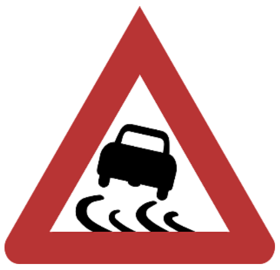
Here are 6 German traffic signs that I found on the web:



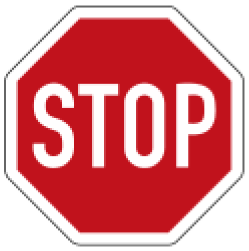
All the images were classified correctly except the “Stop Sign” and “bicycle crossing”.



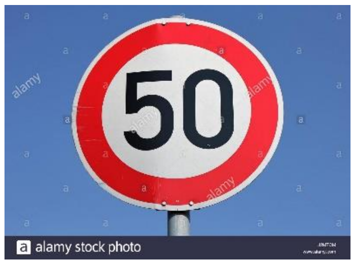
this was classified as →



This is understandable to some extent since they look similar when scaled down. Fine-tuning the model to select the right images for data augmentation might have helped (instead of just random selecting)



this was classified as →



This is not clear. STOp is a distinct sign that can be classifed easily. Interesting to note that “correct” classification is not part of the top 5 probabilties as well.

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).

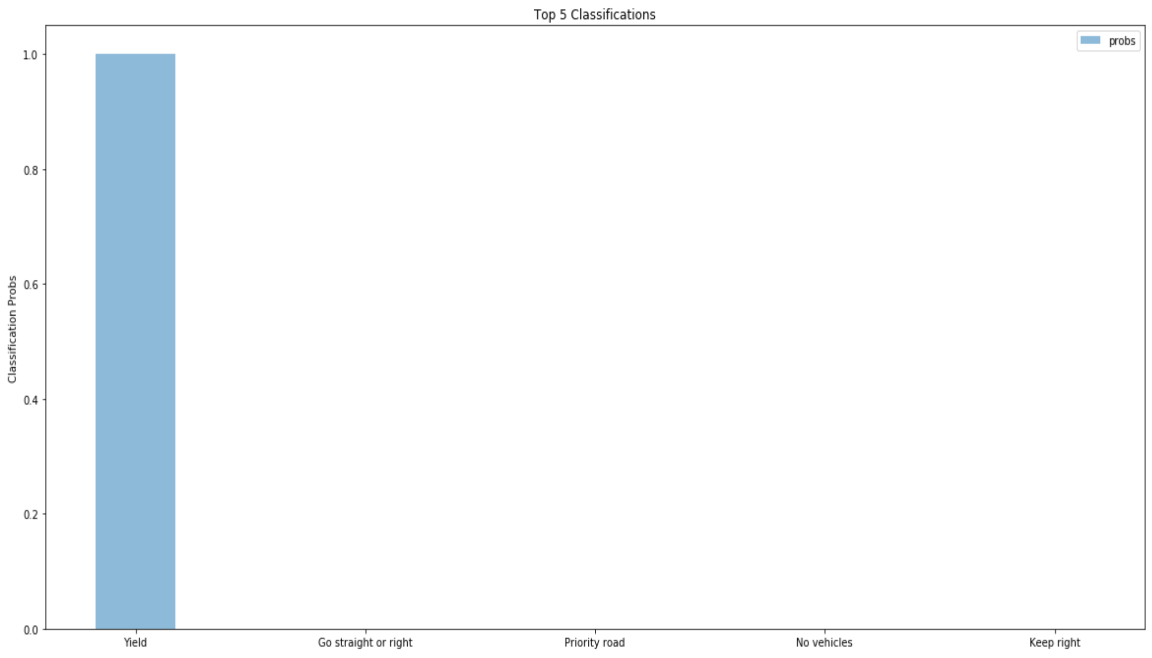
Here are the results of the prediction:

Image	Prediction
Yield	**Yield**
Stop	**Speed limit (50km/h)**
Keep right	**Keep right**
Turn right ahead	**Turn right ahead**
Bicycles crossing	**Slippery road**

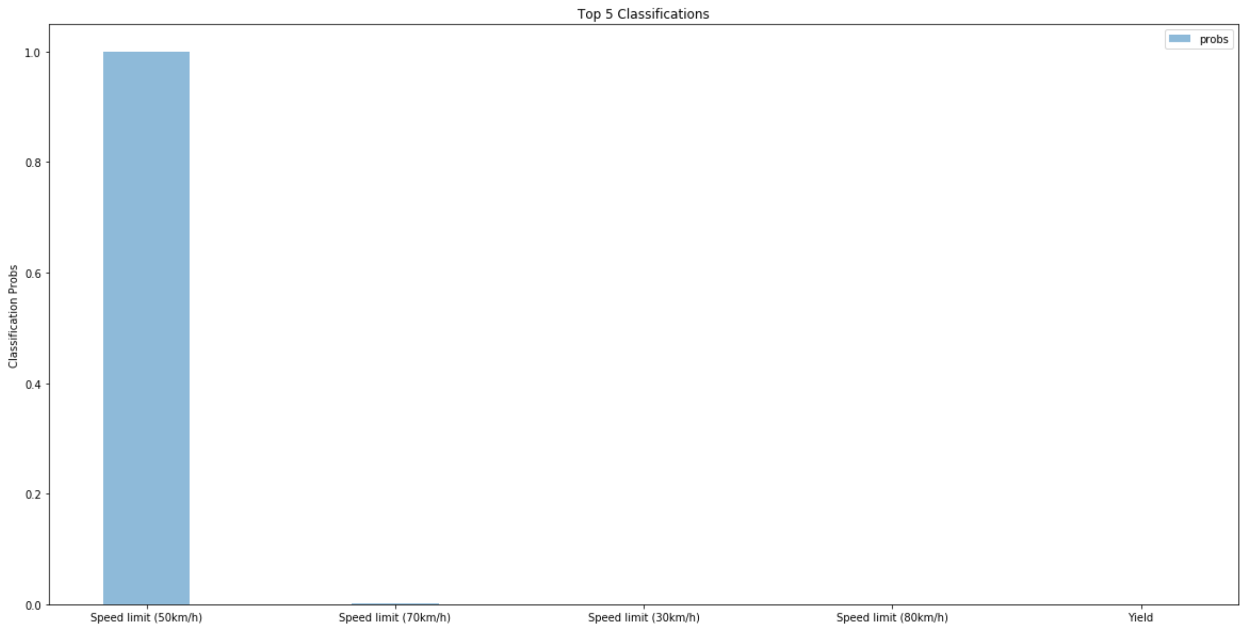
The model was able to correctly guess 3 of the 5 traffic signs, which gives an accuracy of 60%. This compares unfavorably to the accuracy on the test set of 93.2%.

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)

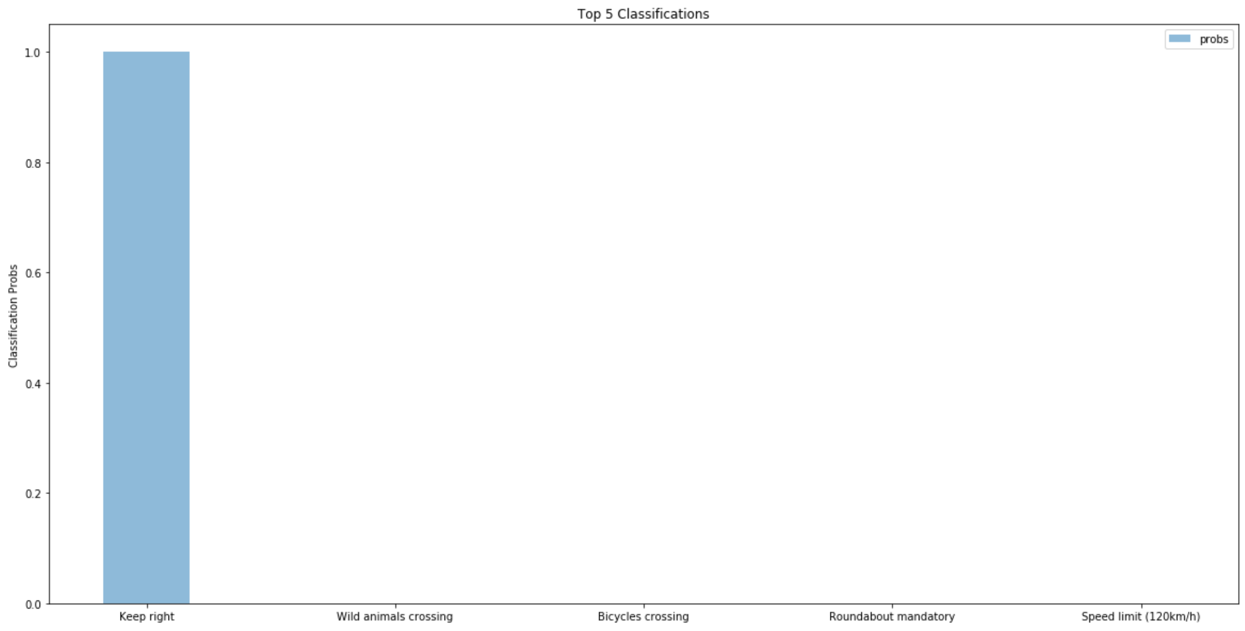
```
image1 probabilities [['1.000', '0.000', '0.000', '0.000', '0.000']]
```



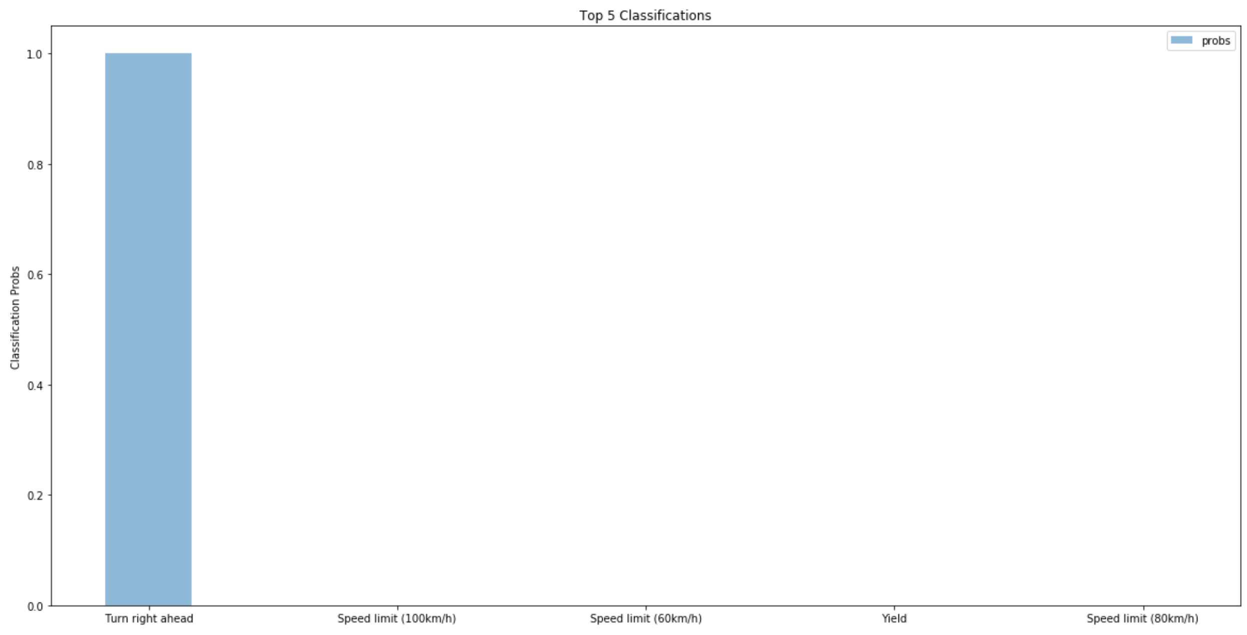
```
image2 probabilities [['0.999', '0.001', '0.000', '0.000', '0.000']]
```



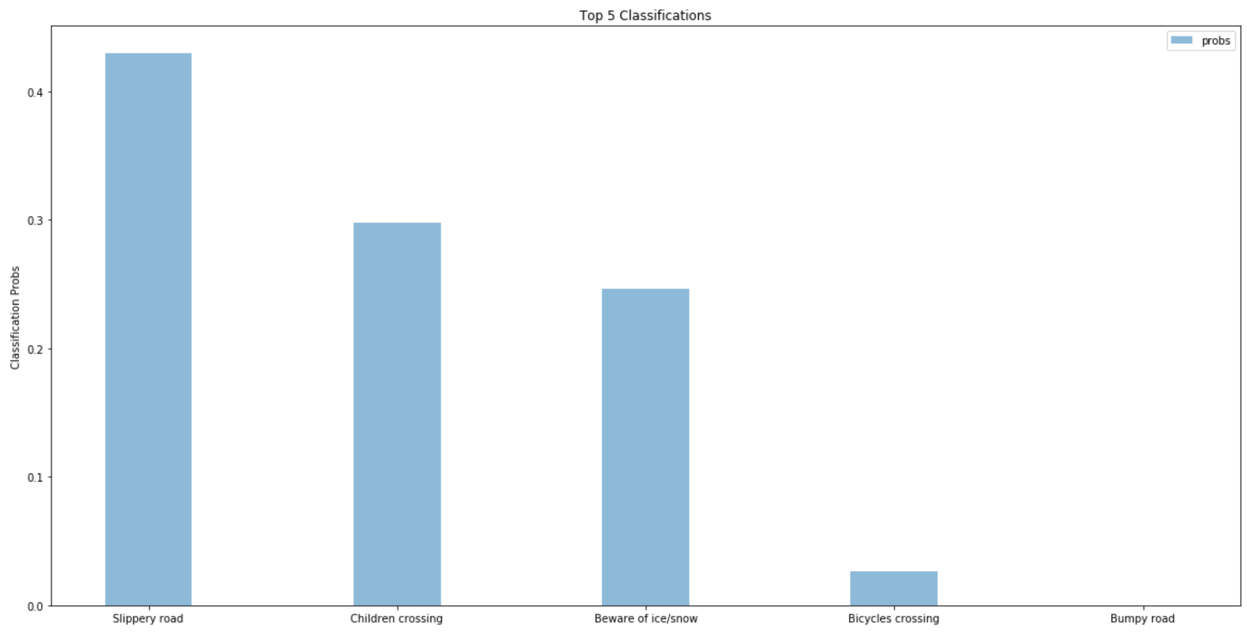
```
image3 probabilities [['1.000', '0.000', '0.000', '0.000', '0.000']]
```



```
image4 probabilities [['1.000', '0.000', '0.000', '0.000', '0.000']]
```

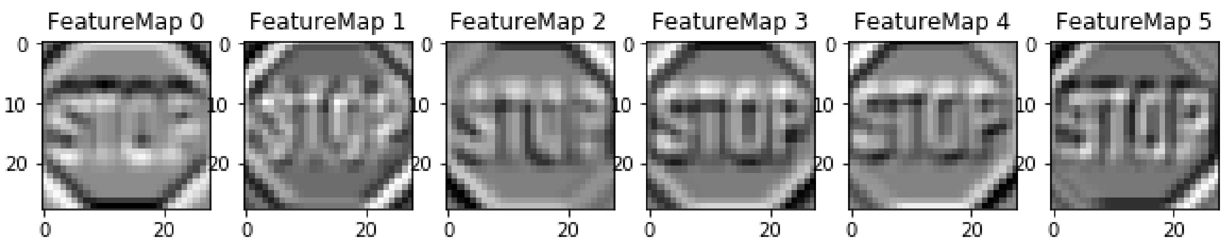


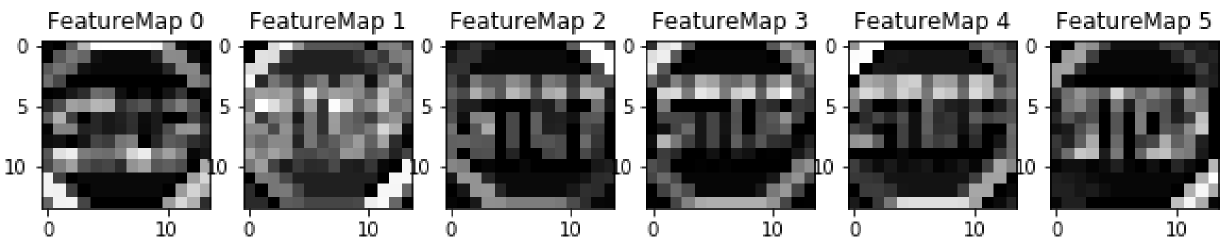
```
image5 probabilities [['0.430', '0.298', '0.246', '0.026', '0.000']]
```

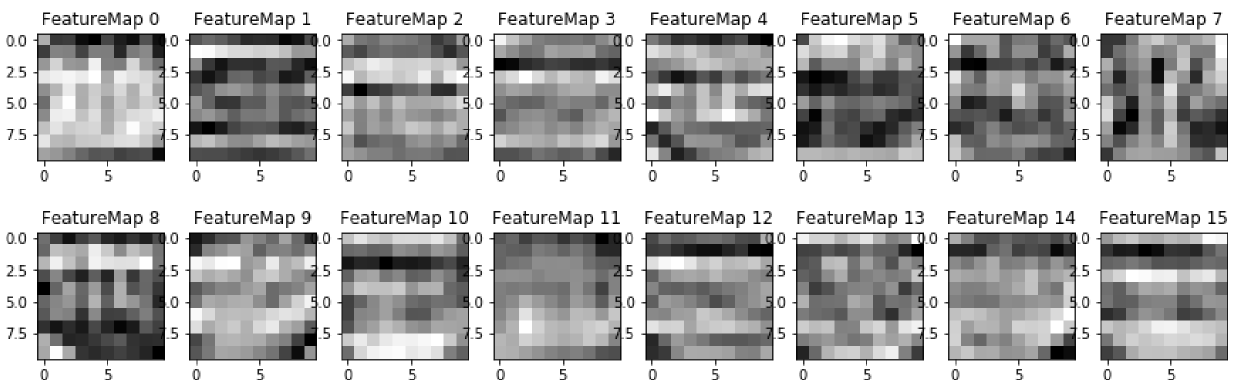


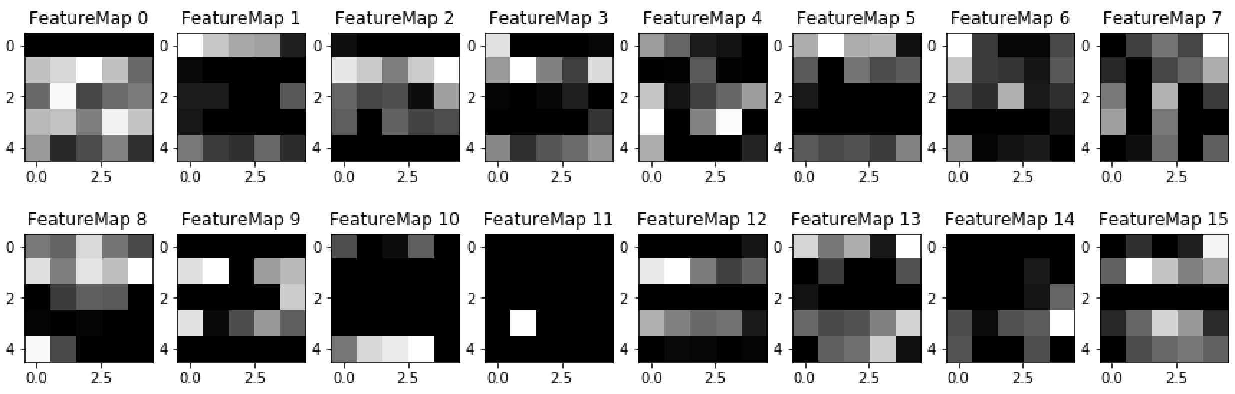
(Optional) Visualizing the Neural Network (See Step 4 of the Ipython notebook for more details)

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


FeatureMap 0
FeatureMap 1
FeatureMap 2
FeatureMap 3
FeatureMap 4
FeatureMap 5


FeatureMap 0
FeatureMap 1
FeatureMap 2
FeatureMap 3
FeatureMap 4
FeatureMap 5


FeatureMap 0
FeatureMap 1
FeatureMap 2
FeatureMap 3
FeatureMap 4
FeatureMap 5
FeatureMap 6
FeatureMap 7
FeatureMap 8
FeatureMap 9
FeatureMap 10
FeatureMap 11
FeatureMap 12
FeatureMap 13
FeatureMap 14
FeatureMap 15


FeatureMap 0
FeatureMap 1
FeatureMap 2
FeatureMap 3
FeatureMap 4
FeatureMap 5
FeatureMap 6
FeatureMap 7
FeatureMap 8
FeatureMap 9
FeatureMap 10
FeatureMap 11
FeatureMap 12
FeatureMap 13
FeatureMap 14
FeatureMap 15