Traffic Sign Recognition

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
- Visualize the Convolutional Neural Network's State with Test Images, plot the feature maps
- 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!

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

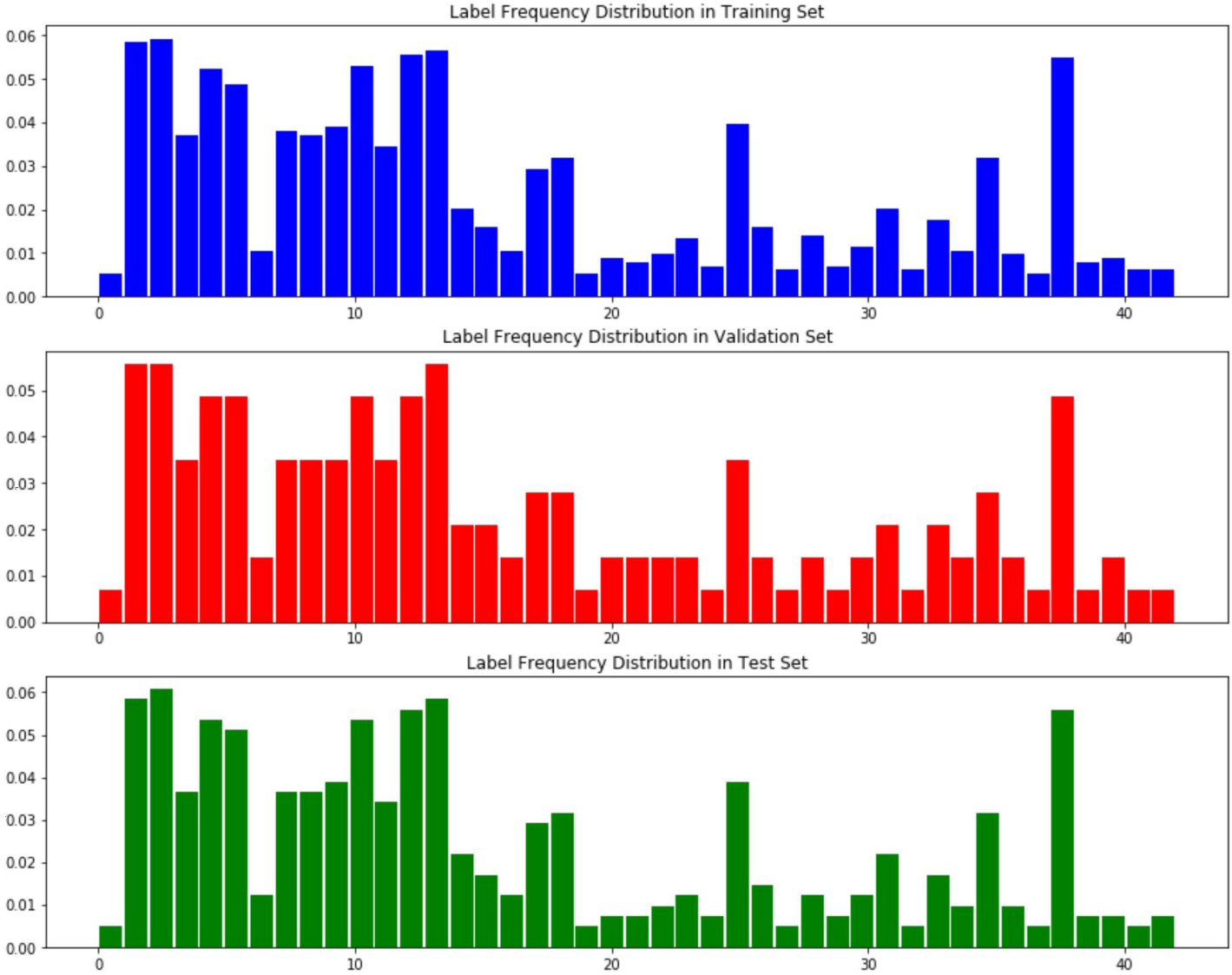
I used the numpy and pandas library to calculate summary statistics of the traffic signs data set:

- The size of training set is 34799 images.
- The size of the validation set is 4410 images.
- The size of test set is 12630 images.

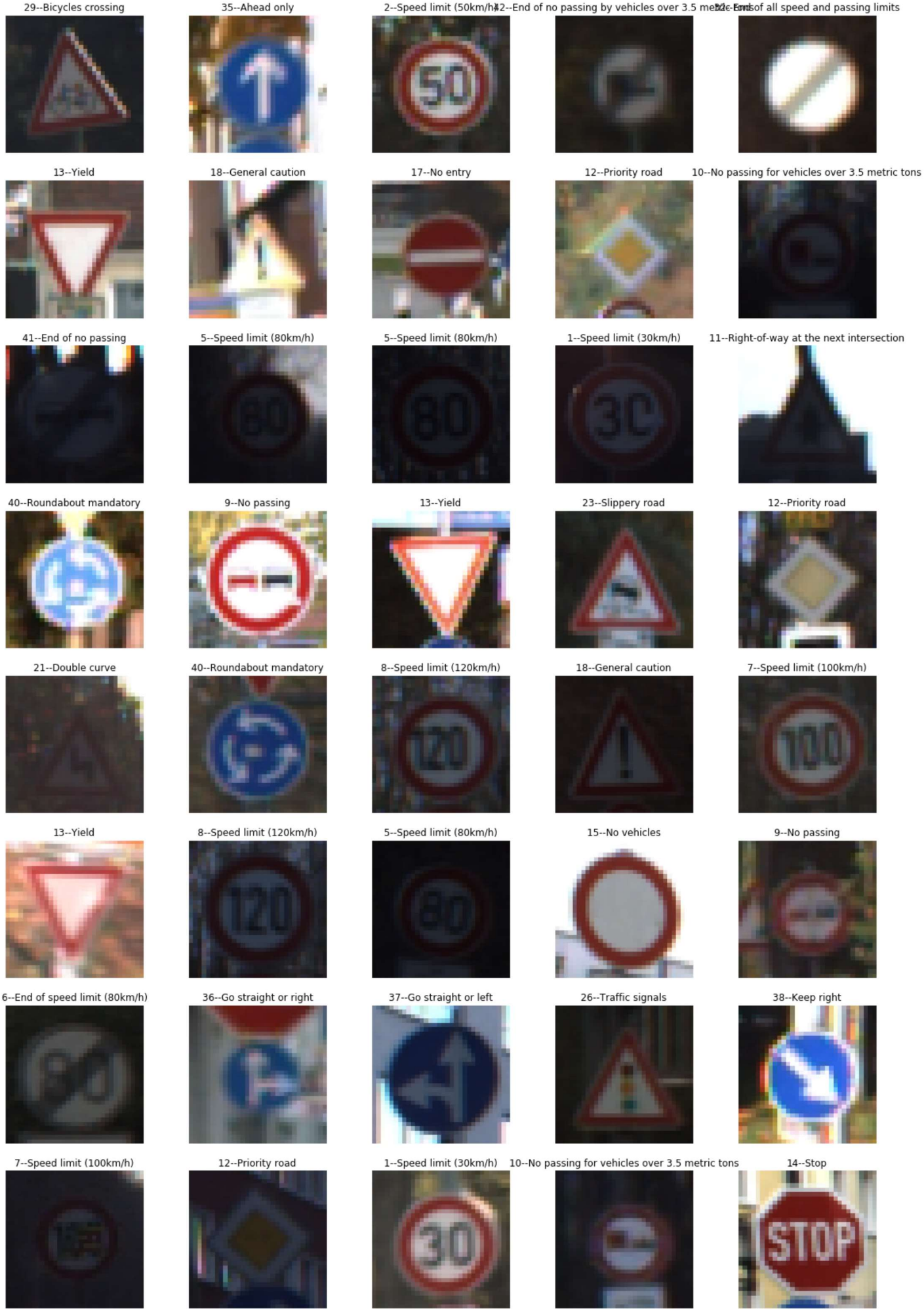
- The shape of a traffic sign image is (32,32,3)
- The number of unique classes/labels in the data set is 43.

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 data is distributed among the 43 classes. We can see that the label frequency distribution in training set, validation set and test set are highly similar. However, the data set is imbalanced among different classes. But my trained model can work well even if the training data set is imbalanced.



I randomly choose 40 images from the training set and plot them here. We can see that there are some images much darker than others. So I'll use histogram equalization technique to tackle this problem.



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.

I use three techniques to preprocess the data set: 1. Grayscale; 2.Histogram Equalization; 3. Mean Normalization.

As a first step, I decided to convert the images to grayscale because many images in the data test are dark and blurred. And the number of images may be not large enough to train a good model from scratch, so dropping the color information may help the model focus more on the edge of signs and the shape information.

Then I use a technique called histogram equalization. Because there are some dark images in the dataset, so I use OpenCV's CLAHE(Contrast Limited Adaptive Histogram Equalization) cv2.createCLAHE() method to improve the contrast of images.

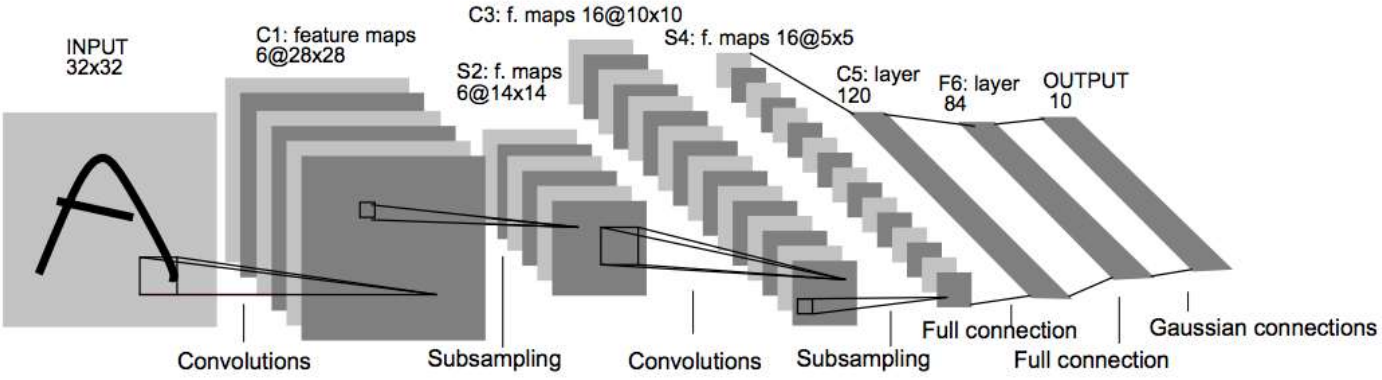
As a last step, I normalized the image data because using mean normalization can make optimizer algorithm more easily minimize the loss function to train the model.

Here is an example of an original image and the preprocessed image:

We can see that the preprocessed images are more clear for recognization than original ones.

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.

I used the famous model--Yan Lecun's LeNet-5 model to finish this task.



My final model consisted of the following layers:

Layer	Description
Input	32x32x1 RGB image
Convolution 5x5	1x1 stride, valid padding, outputs 28x28x6
RELU	----
Max pooling	2x2 stride, outputs 14x14x6
Convolution 5x5	1x1 stride, valid padding, outputs 10x10x16
RELU	----
Max pooling	2x2 stride, outputs 5x5x16
Flatten	outputs 400
Dropout	dropout_keep_probability:0.5
Fully connected	outputs 120
RELU	----
Dropout	dropout_keep_probability:0.5
Fully connected	outputs 84
RELU	----
Dropout	dropout_keep_probability:0.5
Fully connected	outputs 43

Layer	Description
Softmax	predict class probability

In the ipynb file, my final model is in the 15th code cell.

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.

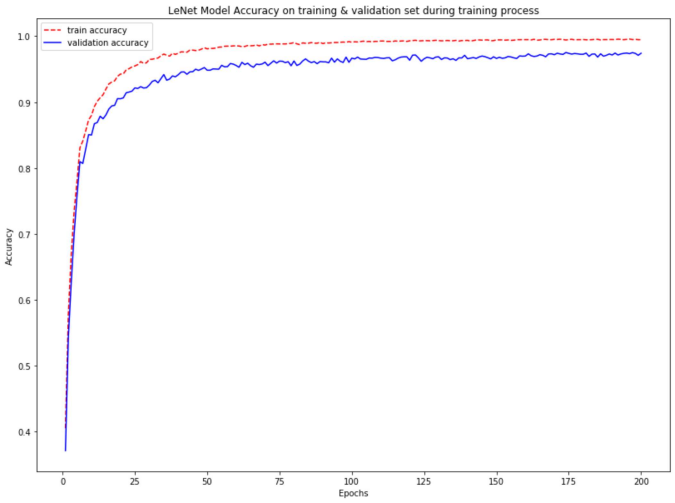
To train the model, I used an Adam Optimizer with 0.0005 as the learning rate. And the number of epochs is 200, the batch size is 128.

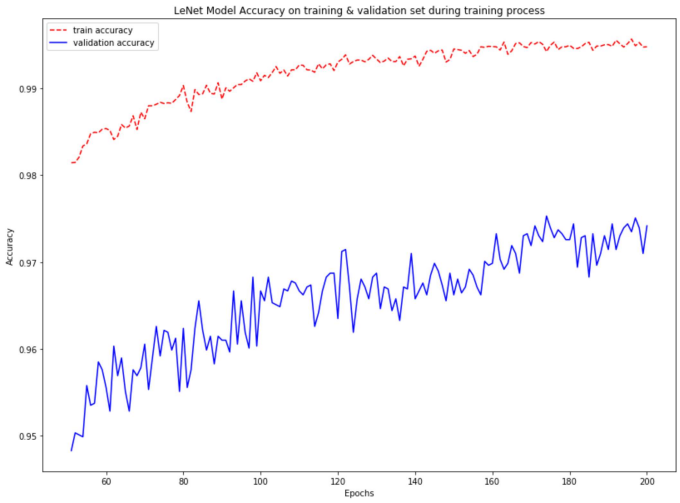
4. Describe the approach taken for finding a solution and getting the validation set accuracy to be at least 0.93. Include in the discussion the results on the training, validation and test sets and where in the code these were calculated. Your approach may have been an iterative process, in which case, 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 set accuracy of 0.995
- validation set accuracy of 0.974
- test set accuracy of 0.952

The LeNet model accuracys on the training and validation set during the training process are shown below.





If a well known architecture was chosen:

- What architecture was chosen?

I choose the well-known LeNet-5 model to finish this traffic sign classification task.

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

  Because we have successfully solved the handwritten digits multi-classification task with this model. And the image in MNIST dataset is 32x32x1, the image in this traffic sign dataset is 32x32x3. After converting the traffic sign image to grayscale, I can use the LeNet-5 model with only a few modifications. Both tasks are similar.

  In order to prevent overfitting, I also insert some dropout layers between the fully connected layers.

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

  I achieved the training set accuracy of 0.995, the validation set accuracy of 0.974, the test set accuracy of 0.952. So we can see this model works very 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 five German traffic signs that I found on the web:

The Speed limit (30km/h) sign and the Road work sign images might be difficult to classify because the training dataset quality is not good, my trained model focus more on edges of the sign shapes than the inner pattern.

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.

Here are the results of the prediction:

Image	Prediction
General caution	General caution
Road work	Road work
Speed limit (30km/h)	Speed limit (30km/h)

Image	Prediction
Keep right	Keep right
Speed limit (60km/h)	Speed limit (60km/h)

The model was able to correctly guess all of the 5 traffic signs, which gives an accuracy of 100%. This compares favorably to the accuracy on the test set of 95.2%.

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.

The code for making predictions on my final model is located in the 24th, 25th, 26th and 27th cell of the Ipython notebook.

For the first image, the model is absolutely sure that this is a General caution sign (probability of 0.999874), and the image does contain a General caution sign. The top five softmax probabilities were

Probability	Prediction
0.999874	General caution
0.000126	Traffic signals
0.000000	Pedestrians
0.000000	Road narrows on the right
0.000000	Right-of-way at the next intersection

For the second image, the model is relatively sure that this is a Road work sign(probability of 0.752628), and the image does contain a Road work sign. The top five softmax probabilities were

Probability	Prediction
0.752628	Road work
0.164122	Bumpy road
0.021112	Wild animals crossing
0.011021	Traffic signals
0.010471	Keep left

For the third image, the model is relatively sure that this is a Speed limit (30km/h) sign(probability of 0.757503), and the image does contain a Speed limit (30km/h) sign. The top five softmax probabilities were

Probability	Prediction
0.757503	Speed limit (30km/h)
0.226876	Speed limit (50km/h)
0.014753	Speed limit (80km/h)
0.000764	Speed limit (60km/h)
0.000096	End of speed limit (80km/h)

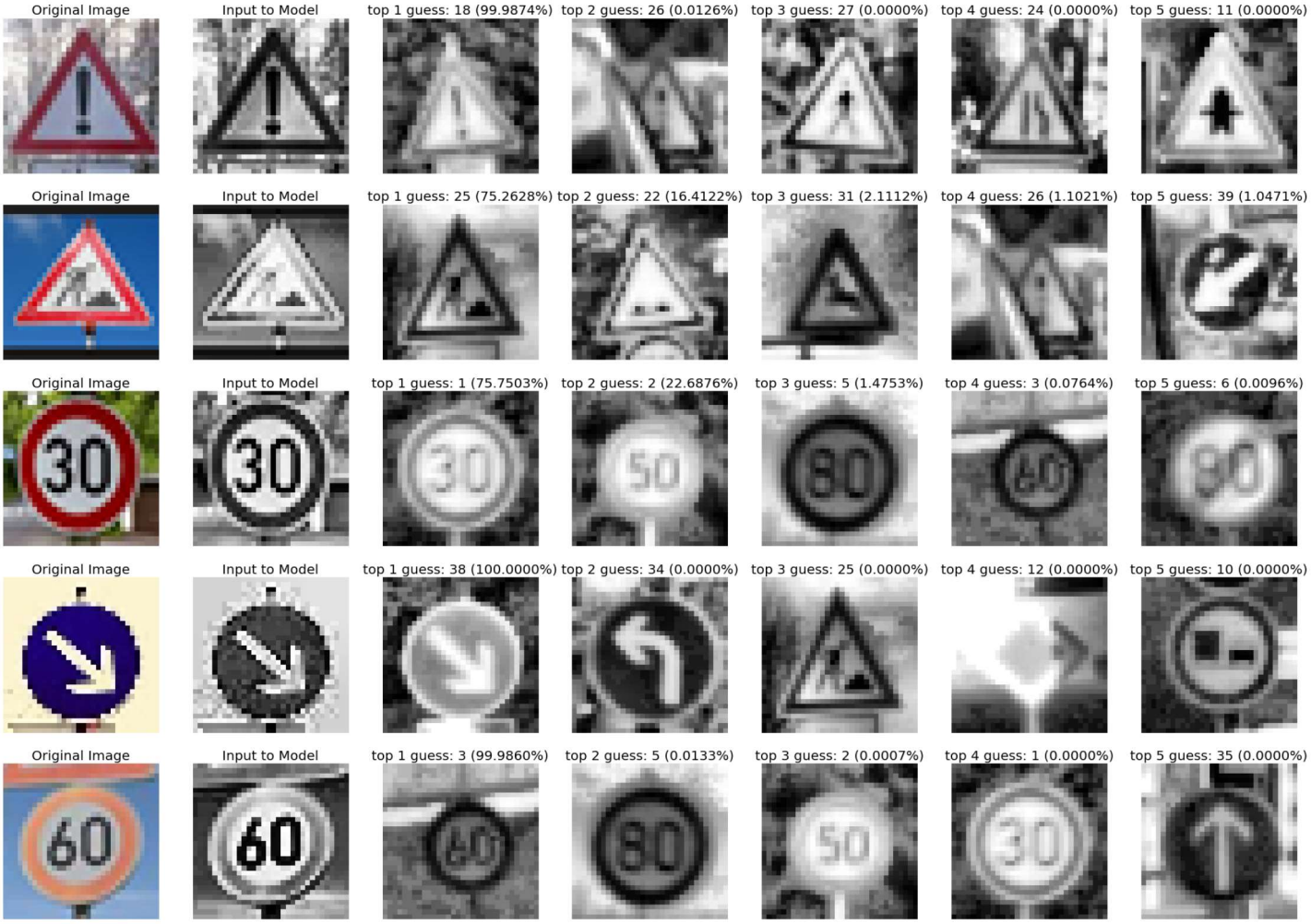
For the fourth image, the model is absolutely sure that this is a Keep right sign(probability of 1.000000), and the image does contain a Keep right sign. The top five softmax probabilities were

Probability	Prediction
1.000000	Keep right
0.000000	Turn left ahead
0.000000	Road work
0.000000	Priority road
0.000000	No passing for vehicles over 3.5 metric tons

For the fifth image, the model is absolutely sure that this is a Speed limit (60km/h) sign(probability of 0.999860), and the image does contain a Speed limit (60km/h) sign. The top five softmax probabilities were

Probability	Prediction
0.999860	Speed limit (60km/h)
0.000133	Speed limit (80km/h)
0.000007	Speed limit (50km/h)
0.000000	Speed limit (30km/h)
0.000000	Ahead only

The image below is the visualization of how certain the model is when predicting on each of the five new images by looking at the softmax probabilities for each prediction.(The columns stand for original image, input to model, and top1-5 guess, respectively.)



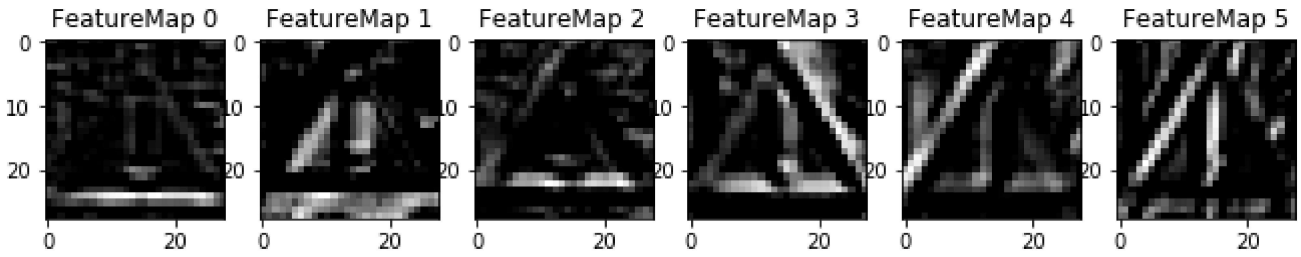
(Optional) Visualizing the Neural Network (See Step 4 of the Ipython notebook for more details)

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

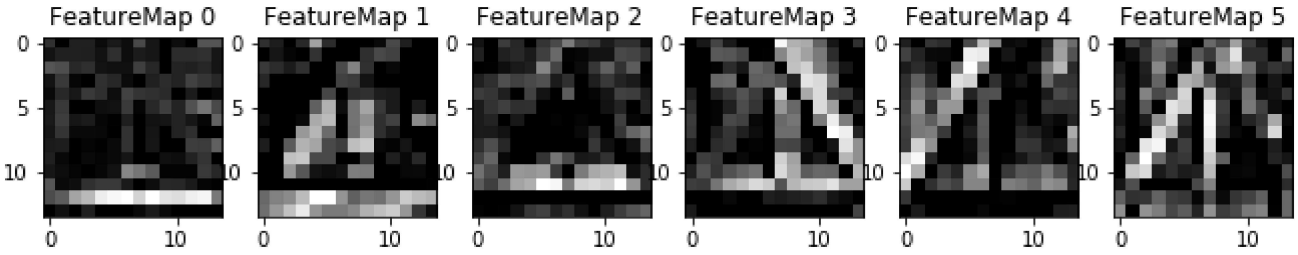
I use the outputFeatureMap() function(in the 28th code cell in .ipynd file) to visualize the LeNet-5 model's convolutional layers' and max pooling layers' feature map.

I use the General caution sign image of the new images as the input stimuli image.

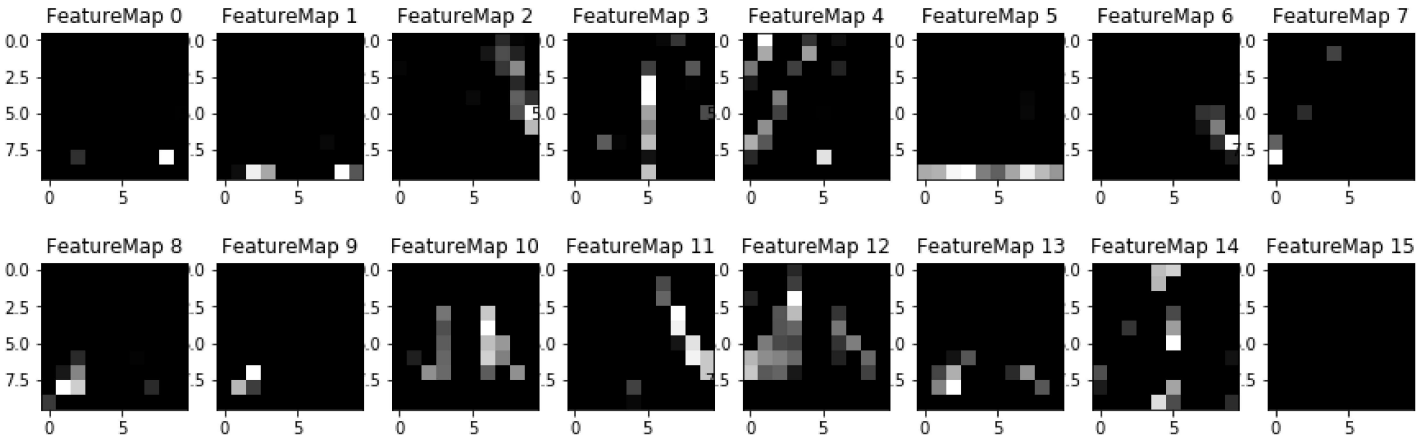
Here are the feature maps of convolutional layer 1. We can see that the network is mainly focusing on the edges of the triangular signs and the inner exclamation mark.



Here are the feature maps of max pooling layer 1. We can see that the max pooling operation reduce the size of the input, and allow the neural network to focus on only the most important elements, in this case, the edges of the signs and the inner exclamation mark.



Here are the feature maps of convolutional layer 2. It is rather hard to determine what the neural network is focusing on in convolutional layer 2 and max pooling layer 2. In these layers, the model is focusing on more abstract and high-level information.



Here are the feature maps of max pooling layer 2: