writeup_template.md

Traffic Sign Recognition

Writeup

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.

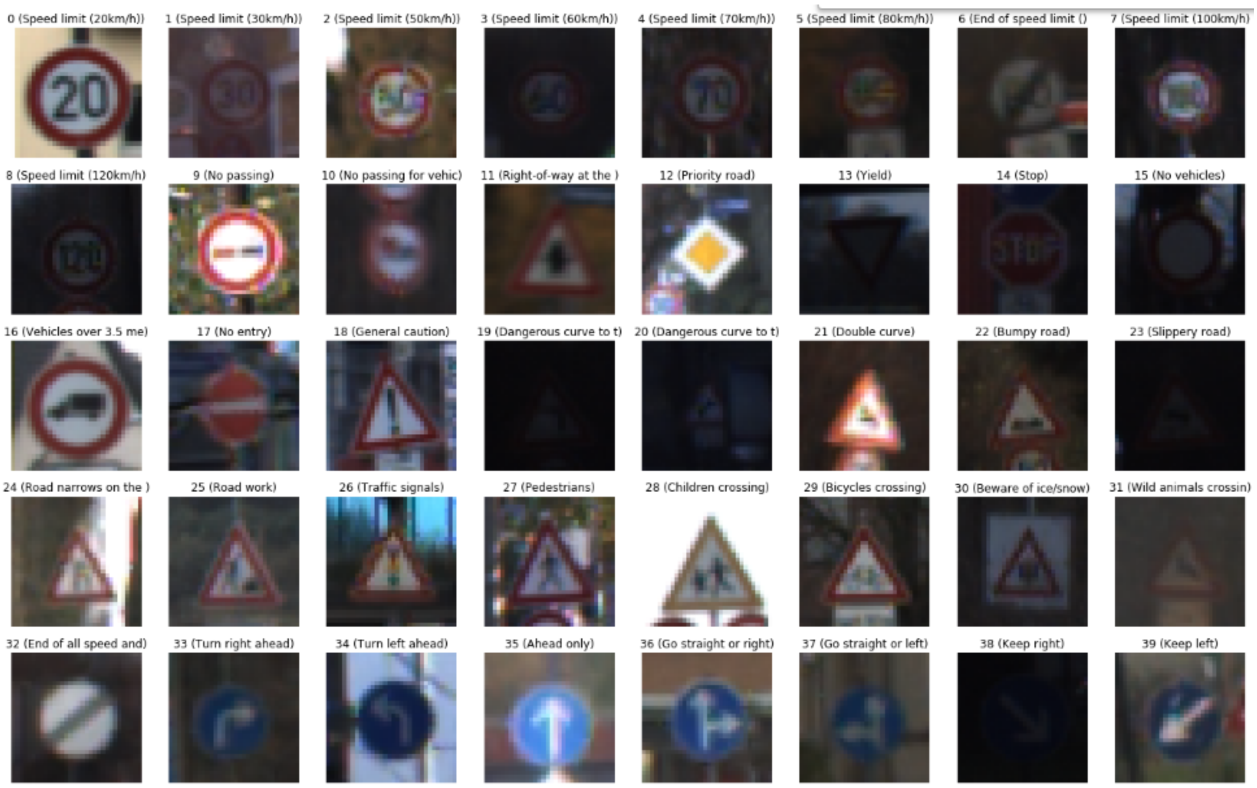
Data Set Summary & Exploration

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

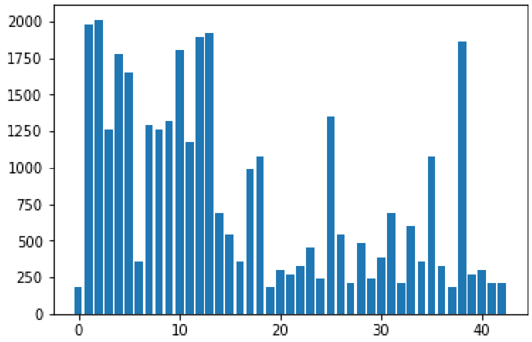
- The size of training set is 34799
- The size of the validation set is 4410
- The size of test set is 12630
- The shape of a traffic sign image is (32,32,3)
- The number of unique classes/labels in the data set is 43

2. Include an exploratory visualization of the dataset.

The figure shows 1 example image for each label in the training set:



Here is an exploratory visualization of the data set. It is a bar chart showing how many samples are contained in the training set per label.

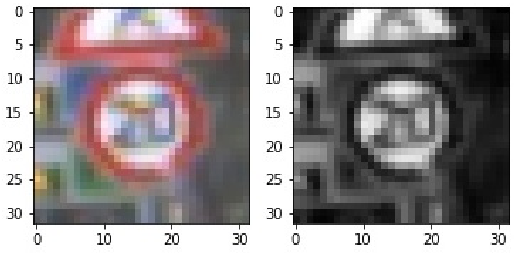


Design and Test a Model Architecture

1. Pre-processing:

As a first step, I decided to convert the images to grayscale because To work on the dataset, it is required to have proper images. To avoid any noise, images were converted to grayscale, which reduced the amount of features; this eventually helped in reducing the execution time. Several research papers have shown very good results when woking with the grayscaled images.

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



Next step was to normalize the image with the help of formula: `(pixel - 128)/ 128` which converted the value of each pixel [0,255] to float with the range [-1,1].

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.

The model architecture is based on the LeNet model architecture. Dropout layers have been added before each fully connected layer in order to prevent overfitting. The final model consisted of the following layers:

The final model consisted of the following layers:

Layer	Description
Input	32x32x1 gray scale 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	
Fully connected	outputs 120
RELU	
Dropout	
Fully connected	outputs 84
RELU	
Dropout	
Fully connected	outputs 43
Softmax	

3. Taining the model

To train the model, I used an Adam optimizer and the following hyperparameters:

- batch size: 128
- number of epochs: 100
- learning rate: 0.0006
- Variables were initialized using the truncated normal distribution with mu = 0.0 and sigma = 0.1
- keep probalbility of the dropout layer: 0.5

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 99.2
- validation set accuracy of 95.5
- test set accuracy of 94.1

5. Solution Approach

An iterative approach for the optimization of validation accuracy was done:

1. Initial model architecture, the original LeNet model from the course was chosen. In order to tailor the architecture for the traffic sign classifier usecase, inputs were adapted so that it accepts the color images from the training set with shape (32,32,3).I modified the number of outputs so that it fits to the 43 unique labels in the training set. The training accuracy was **81.7%** and my test traffic sign "crossing" was not correctly classified. (used hyper parameters: EPOCHS=25, BATCH_SIZE=128, learning_rate = 0.001, mu = 0, sigma = 0.1)

2. After adding the grayscaling preprocessing the validation accuracy increased to **89%** (hyperparameter unmodified)

3. The additional normalization of the training and validation data resulted in a minor increase of validation accuracy: **92%** (hyperparameter unmodified)

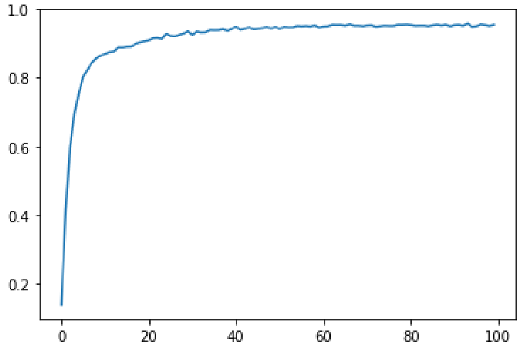
4. reduced learning rate and increased number of epochs. validation accuracy = **93.7%** (EPOCHS = 50, BATCH_SIZE = 128, rate = 0.0007, mu = 0, sigma = 0.1)

5. overfitting. added dropout layer after relu of final fully connected layer: validation accuracy = **94.3%** (EPOCHS = 50, BATCH_SIZE = 128, rate = 0.0007, mu = 0, sigma = 0.1)

6. still overfitting. added dropout after relu of first fully connected layer. Overfitting reduced but still not good

7. added dropout before validation accuracy = 0.953 validation accuracy = **95.1%** (EPOCHS = 75, BATCH_SIZE = 128, rate = 0,0007, mu = 0, sigma = 0.1)

8. further reduction of learning rate and increase of epochs. validation accuracy = **95.5%** (EPOCHS = 100, BATCH_SIZE = 128, rate = 0,0006, mu = 0, sigma = 0.1)



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 few German traffic signs that I found on the web:

alt text

The "bicycles crossing" sign might be difficult to classify because of the distinct pixeled covering the entire canvass in the image. This is quite similar to people crossing, or school nearby signs.

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:

alt text

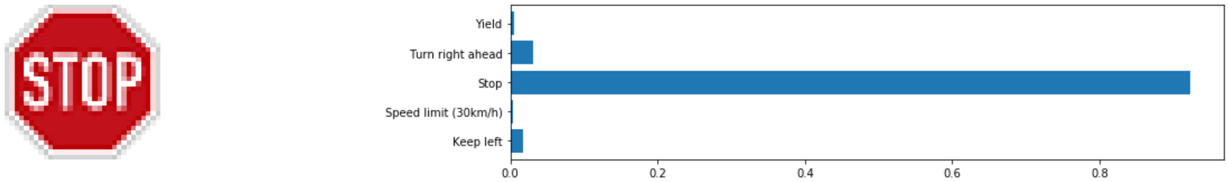
The model was able to correctly guess 9 of the 10 traffic signs, which gives an accuracy of 90%. This compares favorably to the accuracy on the test set of 94.1%.

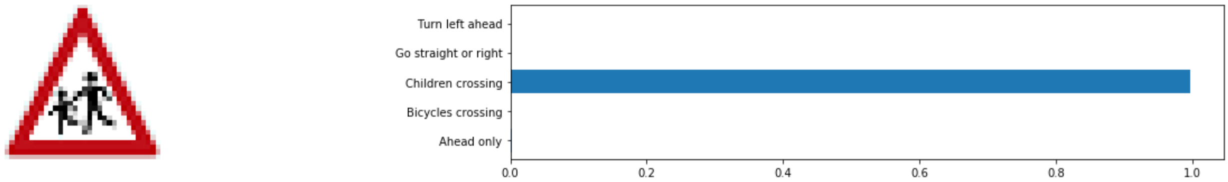
3. Describe how certain the model is when predicting on each of the five new images by looking at the softmax probabilities for each prediction. Provide the top 5 softmax probabilities for each image along with the sign type of each probability. (OPTIONAL: as described in the "Stand Out Suggestions" part of the rubric, visualizations can also be provided such as bar charts)

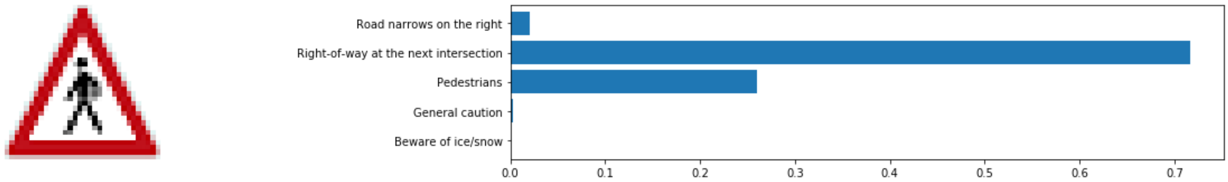
The code for making predictions on my final model is located in the 21th cell of the Jupyter notebook.

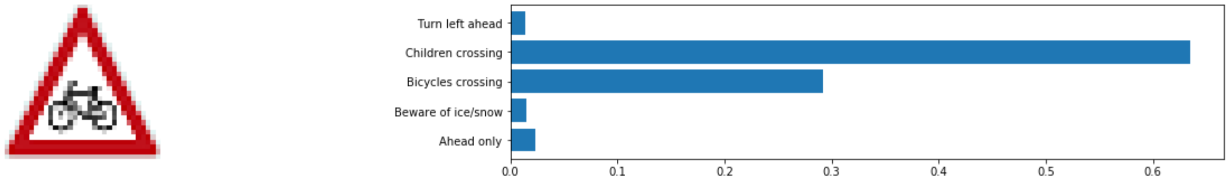
3. Model Certainty - Softmax Probabilities

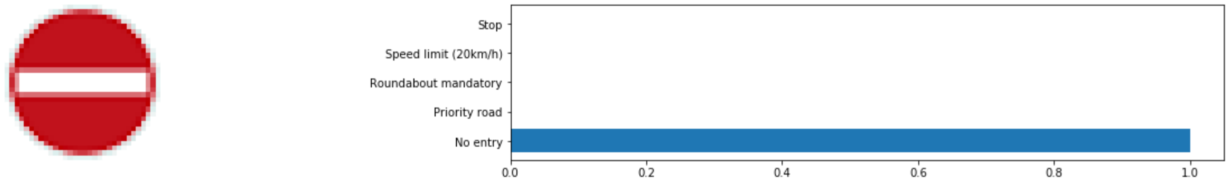
In the following images the top five softmax probabilities of the predictions on the captured images are outputted. As shown in the bar chart the softmax predictions for the correct prediction is bigger than 70% in most cases.

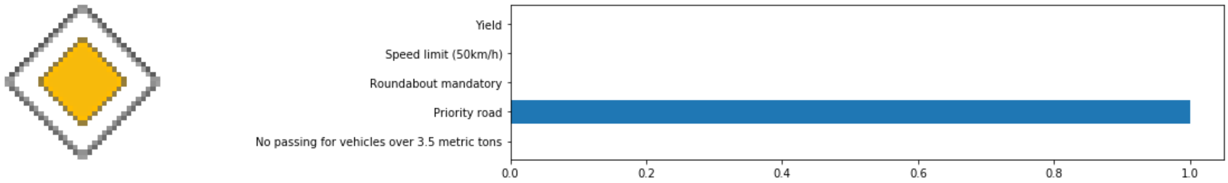


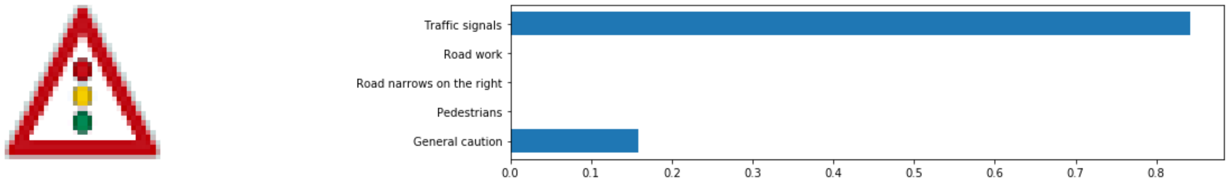


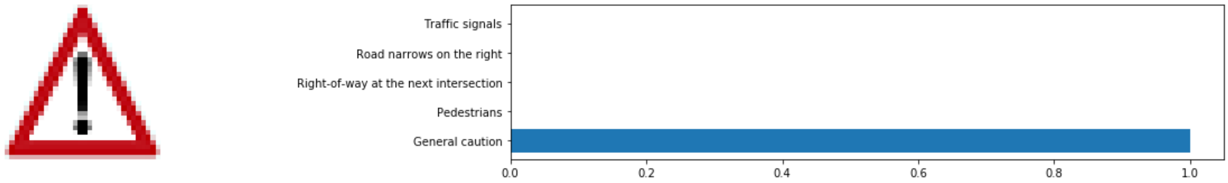


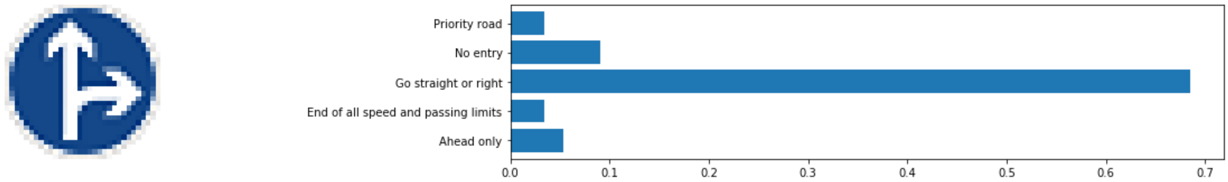


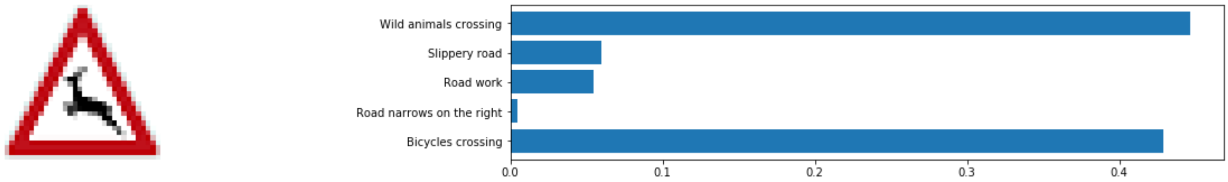




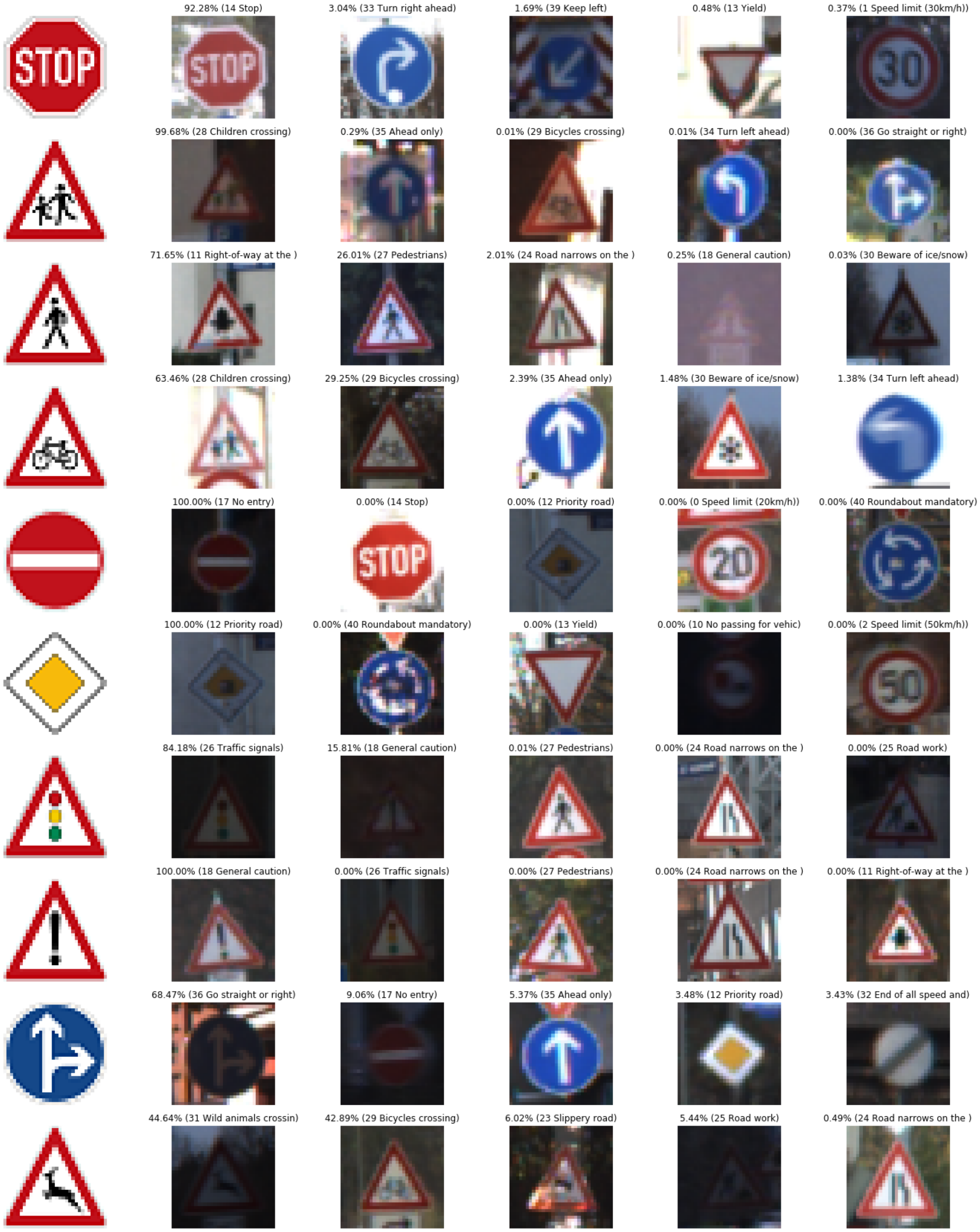








The detailed probabilities and examples of the top five softmax predictions are given in the next image.



(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?

Did not get a chance to work on this given the strict timelines. Will do it once I am free with Term 1