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! here is a link to my 1st version of project code (included hyperlink: https://github.com/SDCND/CarND-Traffic-Sign-Classifier-Project/blob/master/Traffic_Sign_Classifier.ipynb

Here is the link of my 2nd version of project code (included hyperlink: https://github.com/SDCND/CarND-Traffic-Sign-Classifier-Project/blob/master/Traffic_Sign_Classifier2.ipynb

Please consider my solution of review requirements on the end of this document!

Data Set Summary & Exploration

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

The code for this step is contained in the second code cell of the IPython notebook.

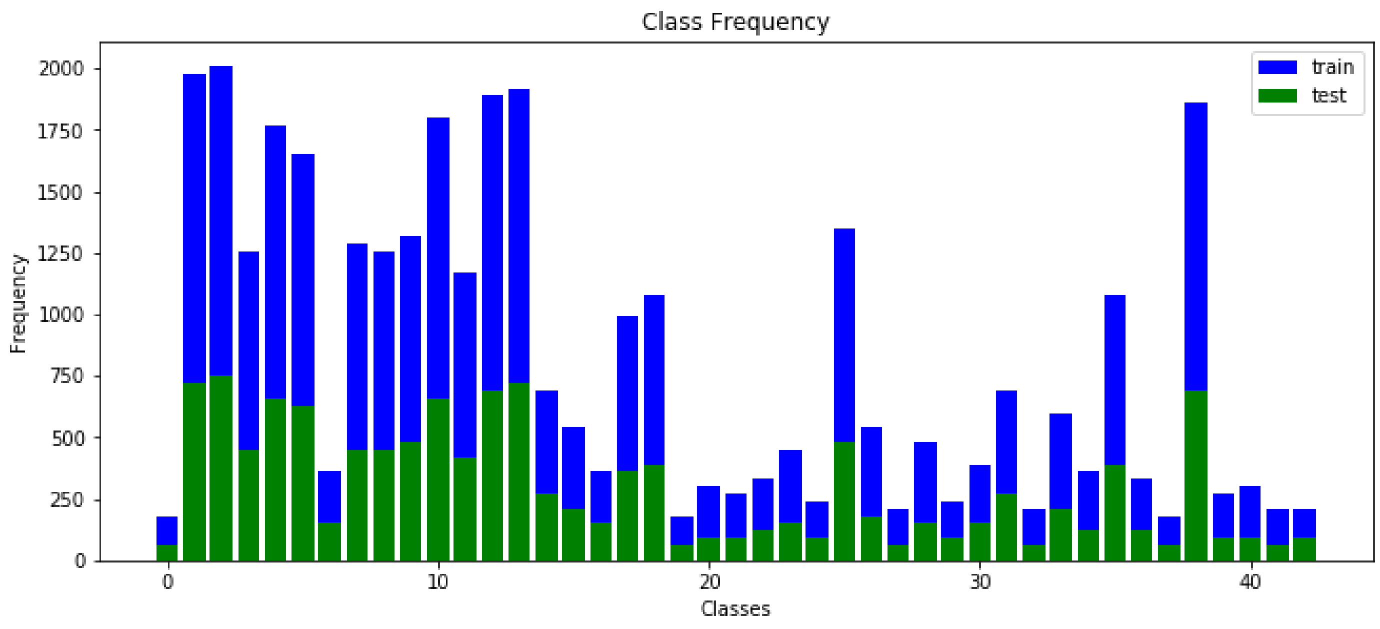
I used the pandas library to calculate summary statistics of the traffic signs data set:

- Number of training examples = 34799
- Number of testing examples = 12630
- Image data shape = (32, 32, 3)
- Number of classes = 43

2. Include an exploratory visualization of the dataset and identify where the code is in your code file.

The code for this step is contained in the third code cell of the IPython notebook.

Here is an exploratory visualization of the data set. It is a bar chart showing how the class frequency is distributed:



Design and Test a Model Architecture

1. Describe how, and identify where in your code, you preprocessed the image data. What tecniques 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.

The code for this step is contained in the fourth code cell of the IPython notebook.

I decided to normalize the images, because it performs better to the training phase. The normalized images are shown in the following steps.

I have not apply the grayscaling yet to see how the neural network will perform the traffic sign recognition without this step.

2. Describe how, and identify where in your code, you set up training, validation and testing data. How much data was in each set? Explain what techniques were used to split the data into these sets. (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, identify where in your code, and provide example images of the additional data)

The code for splitting the data into training and validation sets is contained in the sixth code cell of the IPython notebook.

To cross validate my model, I randomly split the training data into a training set and validation set. I did this by the method train_test_split() from the library sklearn.model_selection.

My final training set had `27839` numbers of images. My validation set had `6960`, test set 20%.

For training the model there is used our training implementation of the convolutional neural network "LeNet" with the parameter initializations:

```
•  EPOCHS = 100
•  BATCH_SIZE = 128
•  mu = 0
•  sigma = 0.1
•  rate = 0.001
```

3. Describe, and identify where in your code, 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 code for my final model is located in the seventh cell of the ipython notebook.

My final model consisted of the following layers:

Layer	Description
Input	32x32x3 RGB image
Layer 1 Convolution 3x3	1x1 stride, same padding, outputs 28x28x6
RELU	

Layer	Description
Max pooling	2x2 stride, outputs 14x14x6
Layer 2 Convolution 3x3	1x1 stride, valid padding, outputs 10x10x16
RELU	
Max pooling	2x2 stride, outputs 5x5x16
Flatten	Outputs 400
Layer 3 Fully connected	Outputs 120
Relu	
Layer 4 Fully connected	Outputs 84
Relu	
Layer 5 Fully connected	Outputs 43 (=n_classes)
Softmax	

4. Describe how, and identify where in your code, 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.

The code for training the model is located in the sixth cell of the ipython notebook.

- `EPOCHS = 100`
- `BATCH_SIZE = 128`
- `mu = 0`
- `sigma = 0.1`
- `rate = 0.001`

```
Output:

Training Set:     27839 samples
Validation Set:   6960 samples
Test Set:         20 %

Training...

EPOCH 1 ...
```

```
Validation Accuracy = 0.561

EPOCH 2 ...
Validation Accuracy = 0.805

EPOCH 3 ...
Validation Accuracy = 0.877

EPOCH 4 ...
Validation Accuracy = 0.914

EPOCH 5 ...
Validation Accuracy = 0.934

EPOCH 6 ...
Validation Accuracy = 0.950

EPOCH 7 ...
Validation Accuracy = 0.952

EPOCH 8 ...
Validation Accuracy = 0.960

EPOCH 9 ...
Validation Accuracy = 0.965

EPOCH 10 ...
Validation Accuracy = 0.968

EPOCH 11 ...
Validation Accuracy = 0.972

EPOCH 12 ...
Validation Accuracy = 0.973

EPOCH 13 ...
Validation Accuracy = 0.974

EPOCH 14 ...
Validation Accuracy = 0.980

…
…

EPOCH 95 ...
Validation Accuracy = 0.990

EPOCH 96 ...
Validation Accuracy = 0.989

EPOCH 97 ...
Validation Accuracy = 0.990

EPOCH 98 ...
Validation Accuracy = 0.991

EPOCH 99 ...
Validation Accuracy = 0.989

EPOCH 100 ...
Validation Accuracy = 0.991

Model saved
Test Accuracy = 0.934
```

5. Describe the approach taken for finding a solution. 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.

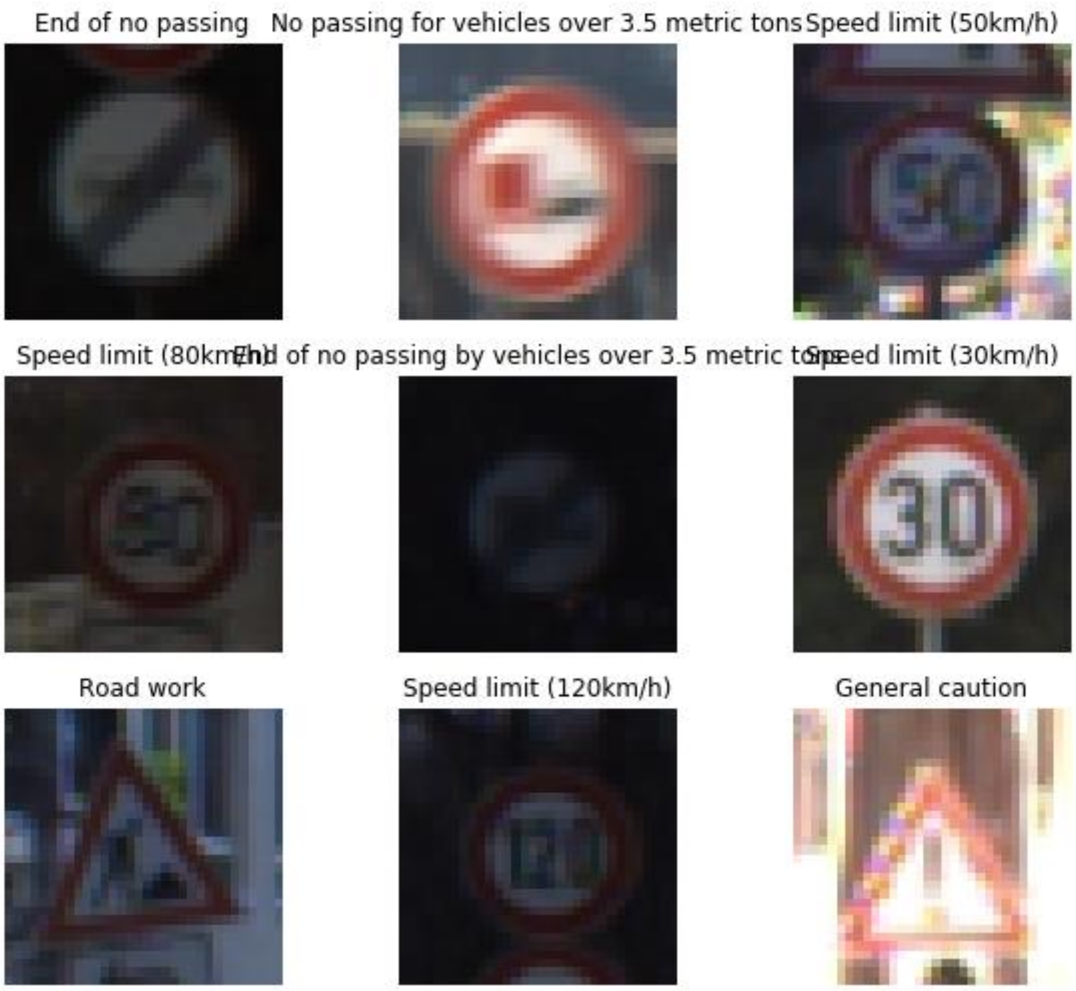
The code for calculating the accuracy of the model is located in the ninth cell of the Ipython notebook.

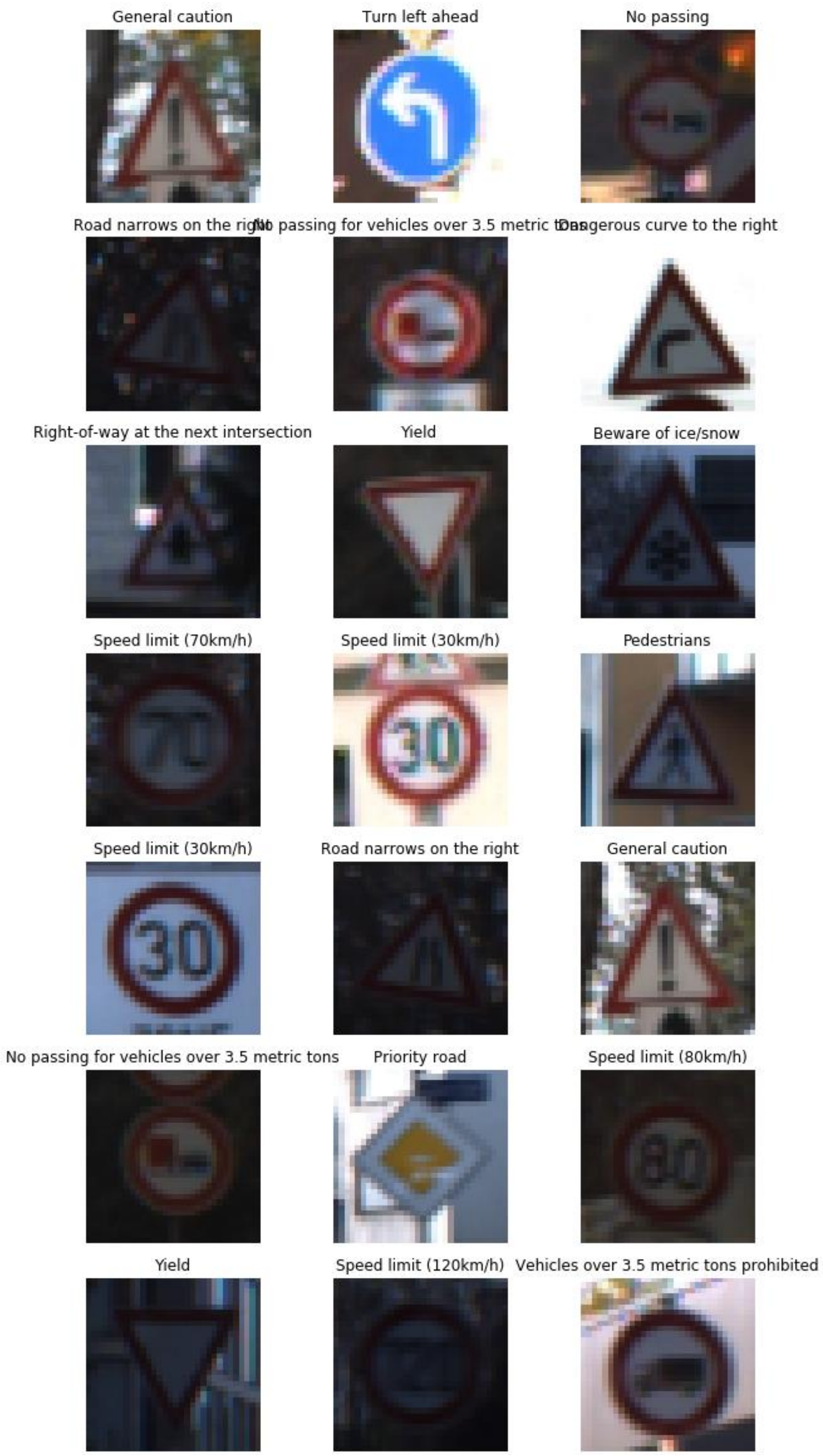
My final model results were:

- validation set accuracy of 0.991
- test set accuracy of 0.934

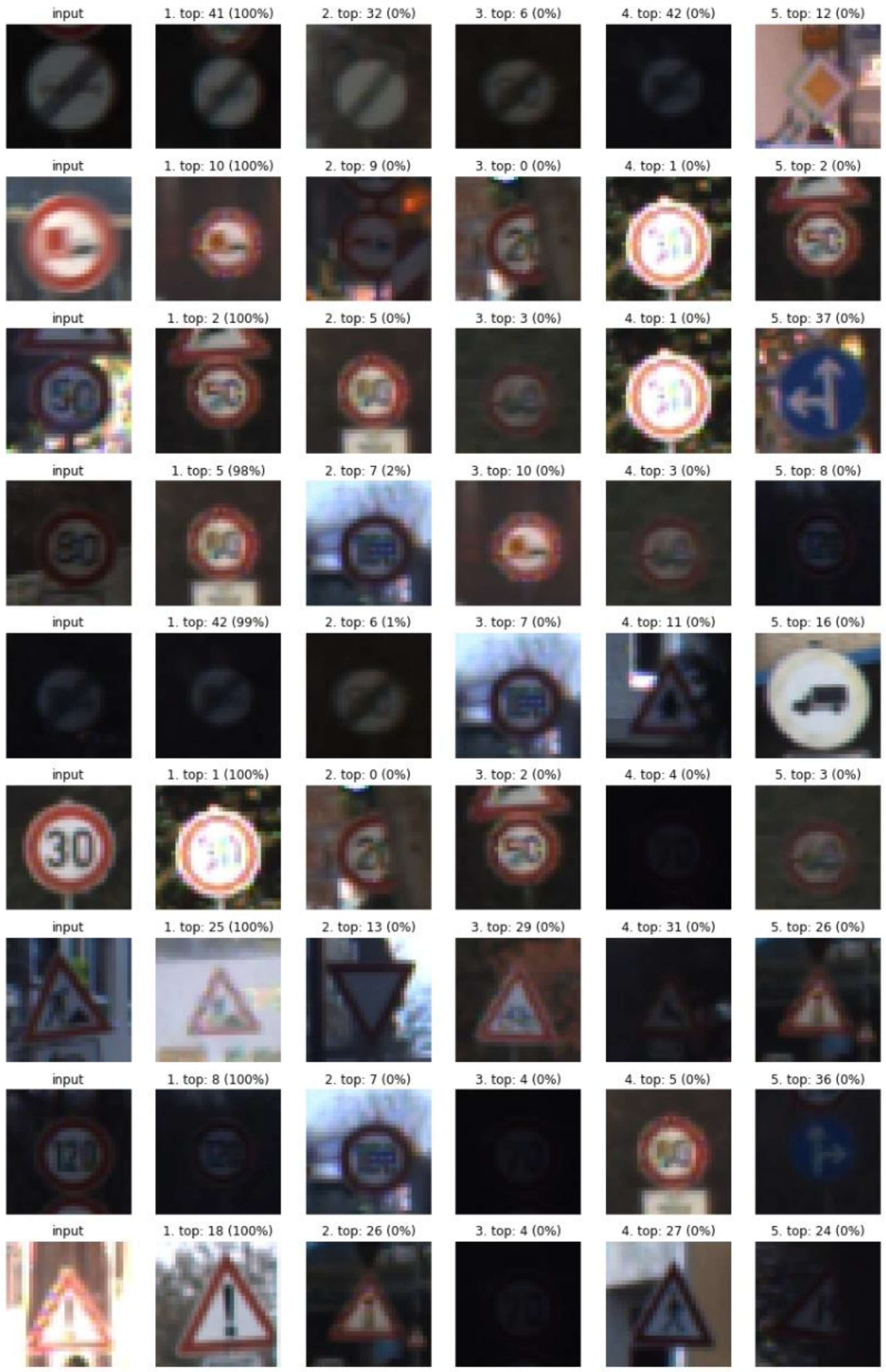
About the training set accuracy:

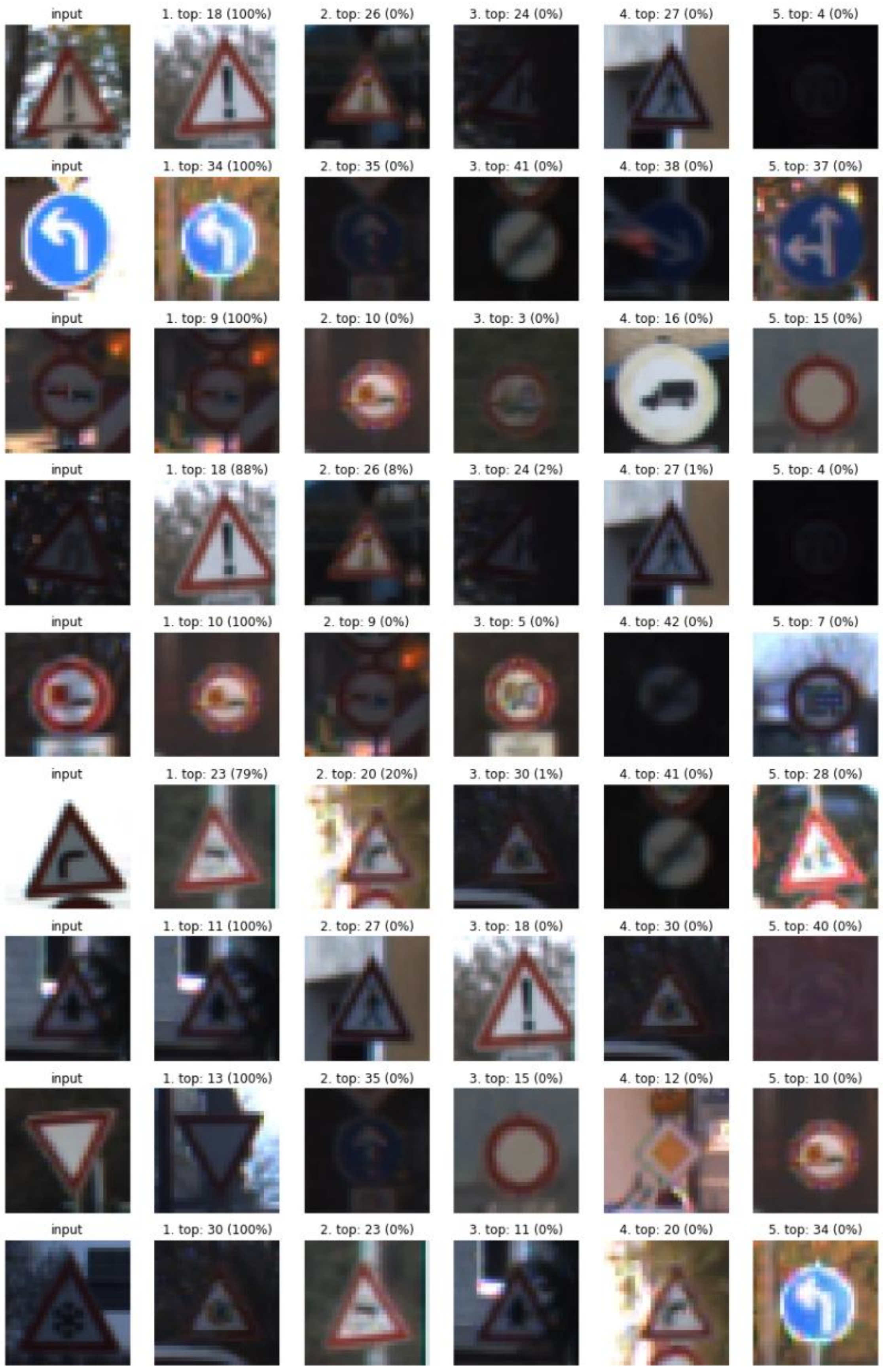
To have a look about the training set accuracy there were randomly chosen the following 30 samples.

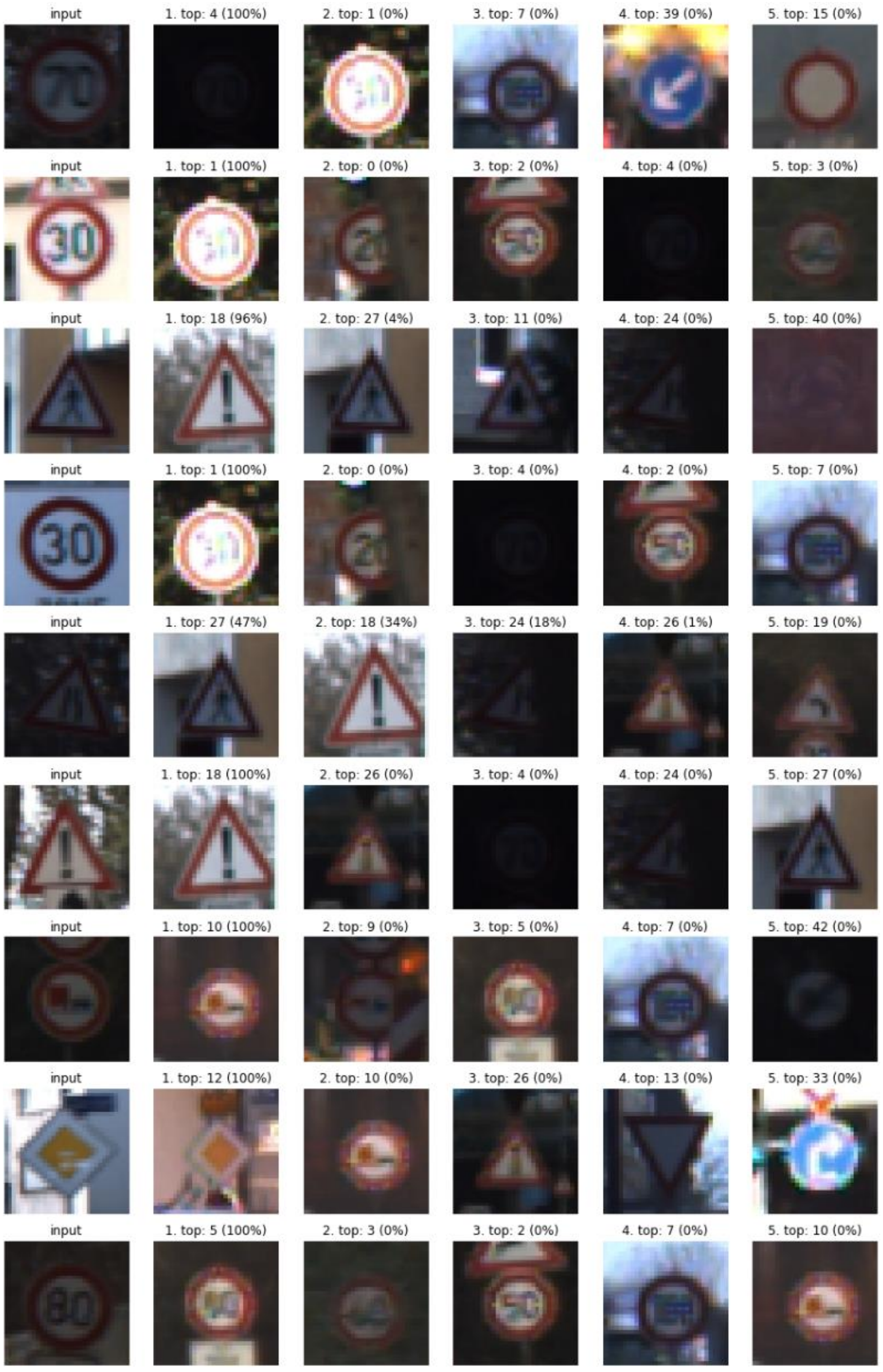



General caution
Turn left ahead
No passing
Road narrows on the right
No passing for vehicles over 3.5 metric tons
Dangerous curve to the right
Right-of-way at the next intersection
Yield
Beware of ice/snow
Speed limit (70km/h)
Speed limit (30km/h)
Pedestrians
Speed limit (30km/h)
Road narrows on the right
General caution
No passing for vehicles over 3.5 metric tons
Priority road
Speed limit (80km/h)
Yield
Speed limit (120km/h)
Vehicles over 3.5 metric tons prohibited

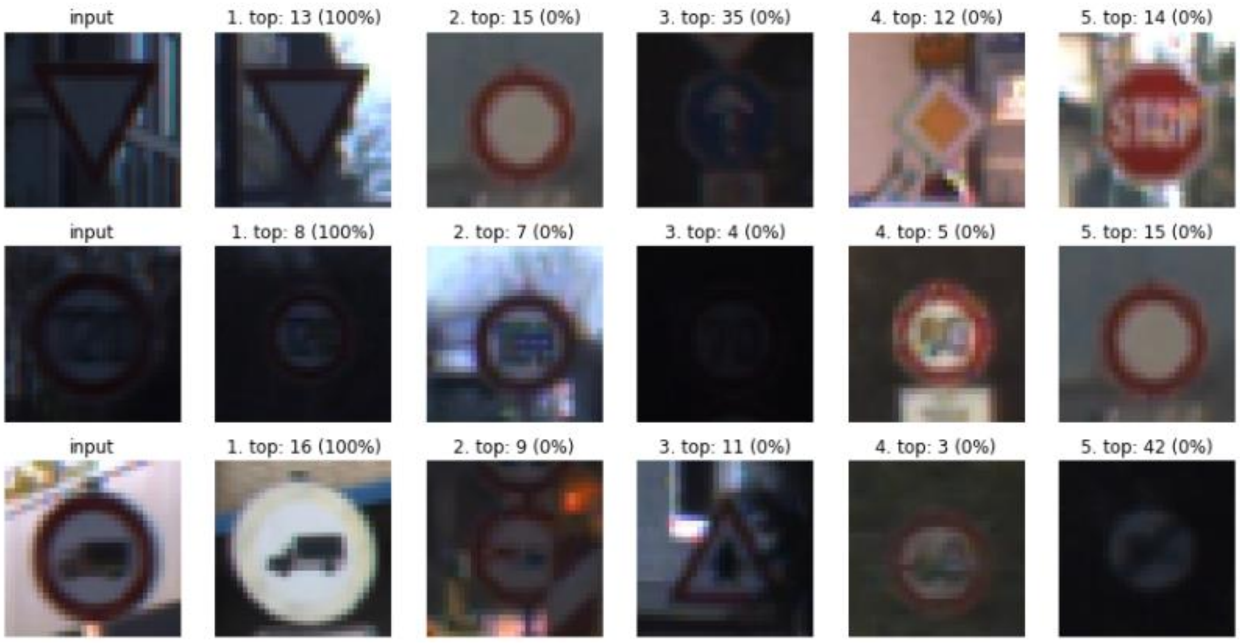
For the prediction of the sign type of each sample there has been determined the top 5 for each sign:




input
1. top: 18 (100%)
2. top: 26 (0%)
3. top: 24 (0%)
4. top: 27 (0%)
5. top: 4 (0%)
input
1. top: 34 (100%)
2. top: 35 (0%)
3. top: 41 (0%)
4. top: 38 (0%)
5. top: 37 (0%)
input
1. top: 9 (100%)
2. top: 10 (0%)
3. top: 3 (0%)
4. top: 16 (0%)
5. top: 15 (0%)
input
1. top: 18 (88%)
2. top: 26 (8%)
3. top: 24 (2%)
4. top: 27 (1%)
5. top: 4 (0%)
input
1. top: 10 (100%)
2. top: 9 (0%)
3. top: 5 (0%)
4. top: 42 (0%)
5. top: 7 (0%)
input
1. top: 23 (79%)
2. top: 20 (20%)
3. top: 30 (1%)
4. top: 41 (0%)
5. top: 28 (0%)
input
1. top: 11 (100%)
2. top: 27 (0%)
3. top: 18 (0%)
4. top: 30 (0%)
5. top: 40 (0%)
input
1. top: 13 (100%)
2. top: 35 (0%)
3. top: 15 (0%)
4. top: 12 (0%)
5. top: 10 (0%)
input
1. top: 30 (100%)
2. top: 23 (0%)
3. top: 11 (0%)
4. top: 20 (0%)
5. top: 34 (0%)



input
1. top: 4 (100%)
2. top: 1 (0%)
3. top: 7 (0%)
4. top: 39 (0%)
5. top: 15 (0%)
input
1. top: 1 (100%)
2. top: 0 (0%)
3. top: 2 (0%)
4. top: 4 (0%)
5. top: 3 (0%)
input
1. top: 18 (96%)
2. top: 27 (4%)
3. top: 11 (0%)
4. top: 24 (0%)
5. top: 40 (0%)
input
1. top: 1 (100%)
2. top: 0 (0%)
3. top: 4 (0%)
4. top: 2 (0%)
5. top: 7 (0%)
input
1. top: 27 (47%)
2. top: 18 (34%)
3. top: 24 (18%)
4. top: 26 (1%)
5. top: 19 (0%)
input
1. top: 18 (100%)
2. top: 26 (0%)
3. top: 4 (0%)
4. top: 24 (0%)
5. top: 27 (0%)
input
1. top: 10 (100%)
2. top: 9 (0%)
3. top: 5 (0%)
4. top: 7 (0%)
5. top: 42 (0%)
input
1. top: 12 (100%)
2. top: 10 (0%)
3. top: 26 (0%)
4. top: 13 (0%)
5. top: 33 (0%)
input
1. top: 5 (100%)
2. top: 3 (0%)
3. top: 2 (0%)
4. top: 7 (0%)
5. top: 10 (0%)



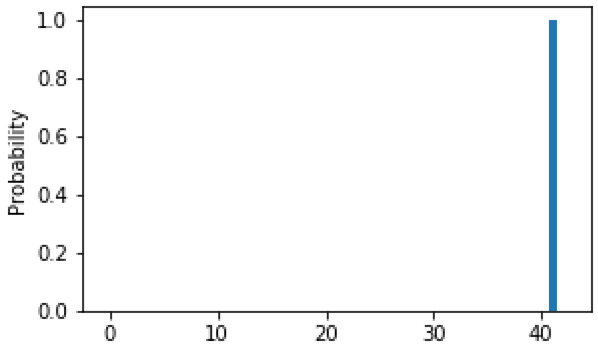
The well-known architecture "LeNet" was chosen. I think, it would be relevant to the traffic sign application, because it is used to classifies digits in digitized and is applied by several banks to recognize hand-written numbers on checks digitized in 32x32 pixel images. The ability to process higher resolution images requires larger and more convolutional layers, so this technique is constrained by the limited availability of computing resources.

Test a Model on New Images

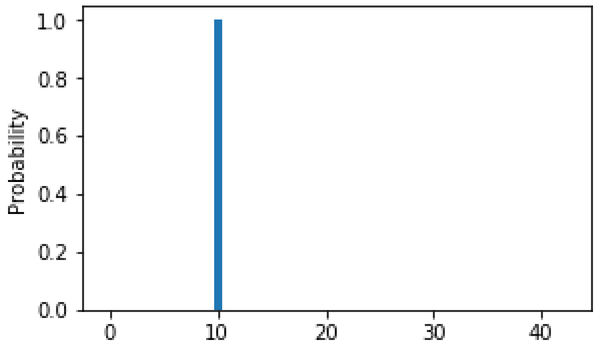
1. Choose five German traffic signs found on the web and provide them in the report. For each image, discuss what quality or qualities might be difficult to classify.

The tested traffic signs were previous shown. Following is listed the recognition performance of each sample sign:

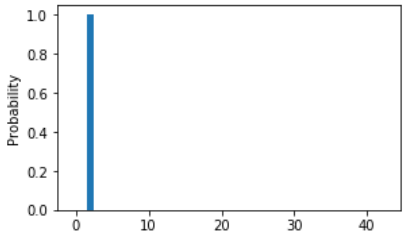
End of no passing



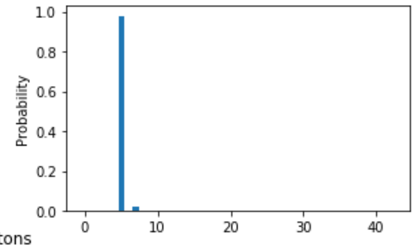
No passing for vehicles over 3.5 metric tons



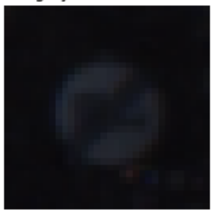
Speed limit (50km/h)

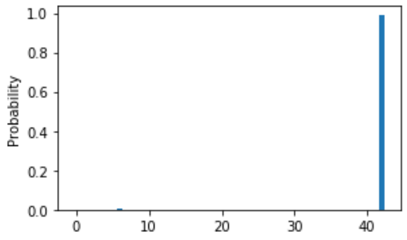


Speed limit (80km/h)

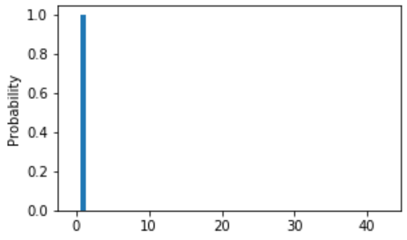


End of no passing by vehicles over 3.5 metric tons

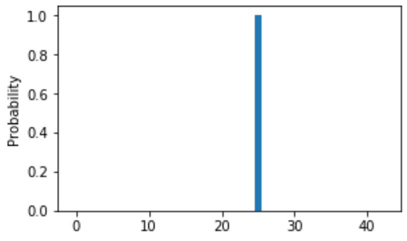




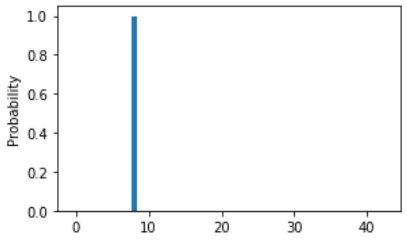
Speed limit (30km/h)



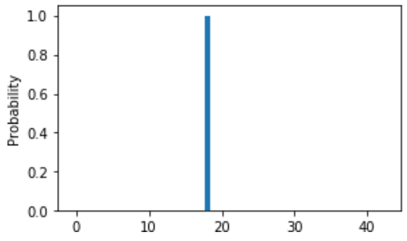
Road work



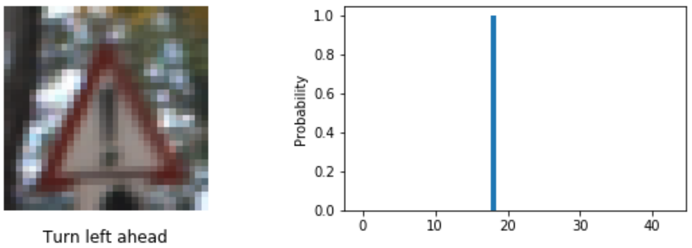
Speed limit (120km/h)



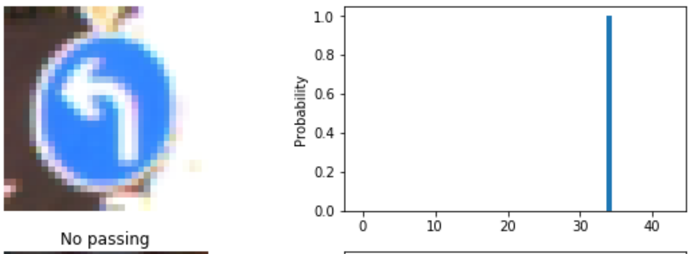
General caution



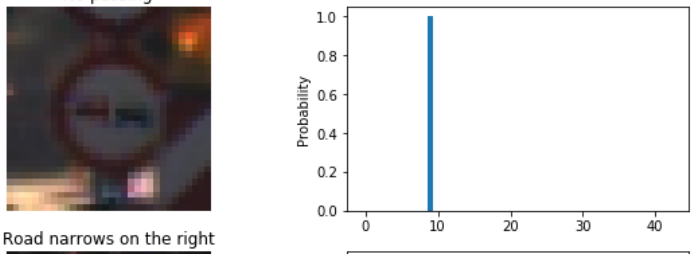
General caution



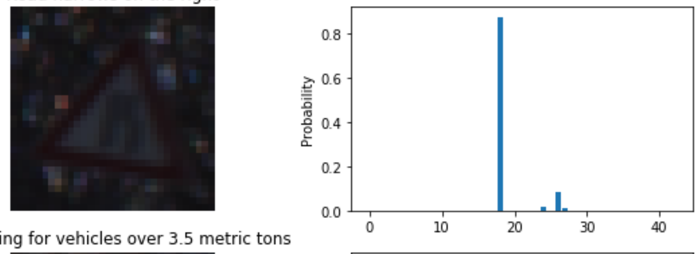
Turn left ahead



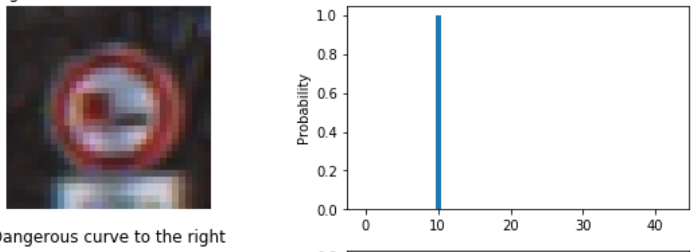
No passing



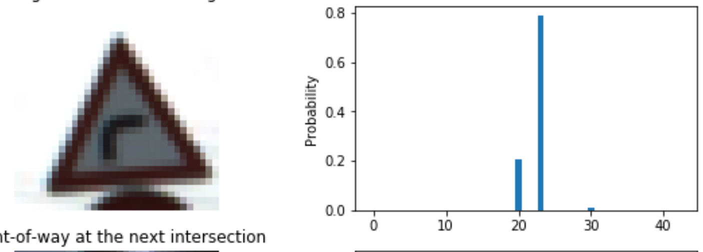
Road narrows on the right



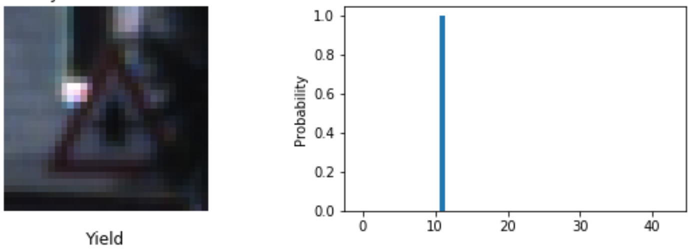
No passing for vehicles over 3.5 metric tons



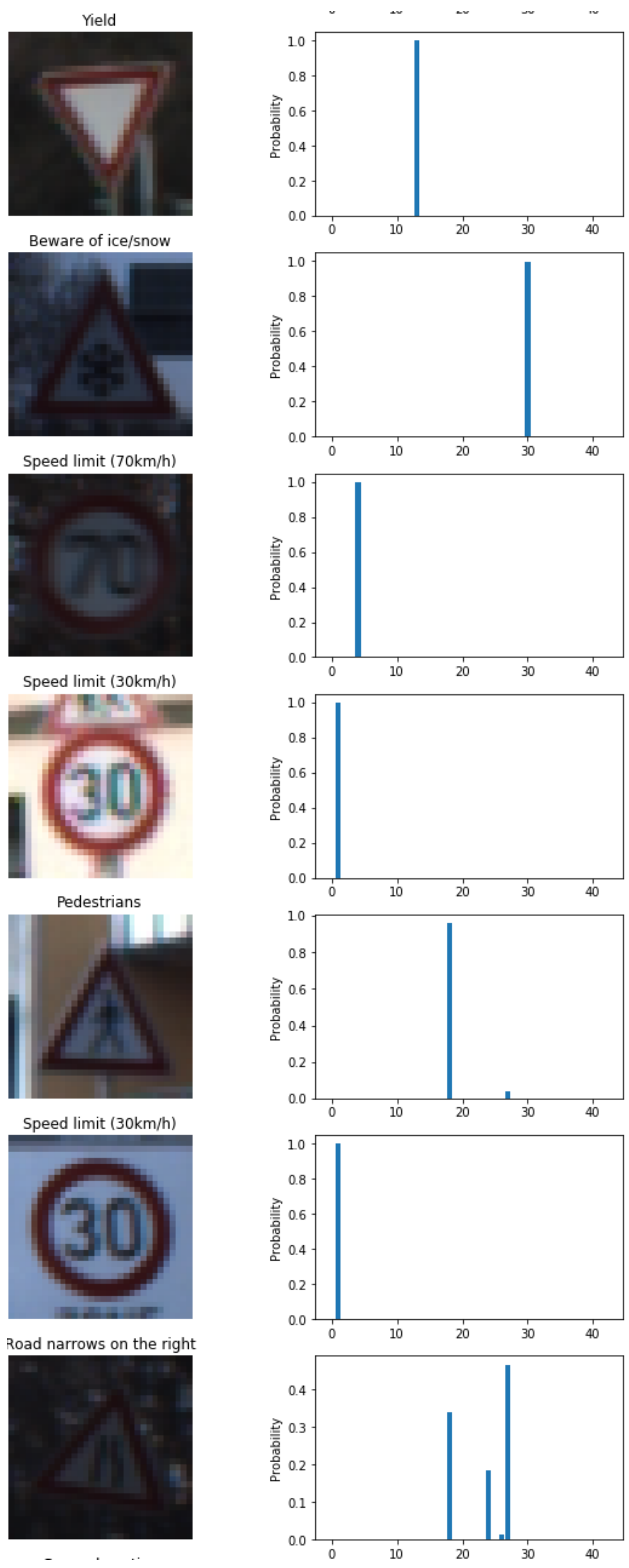
Dangerous curve to the right



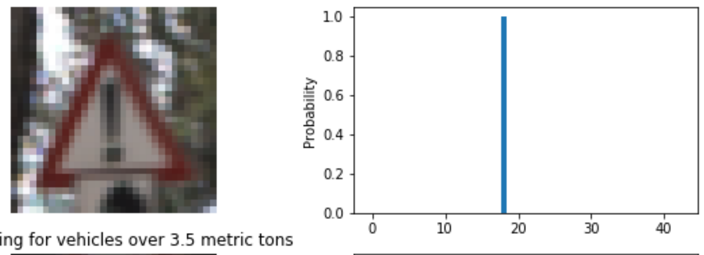
Right-of-way at the next intersection



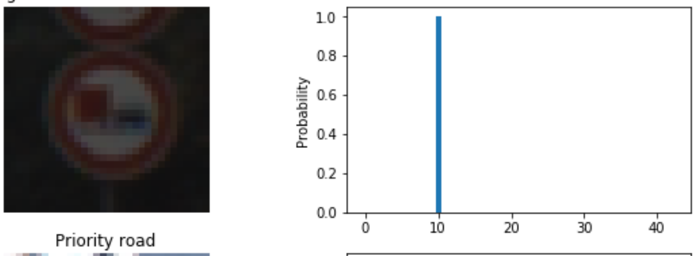
Yield


Yield
Beware of ice/snow
Speed limit (70km/h)
Speed limit (30km/h)
Pedestrians
Speed limit (30km/h)
Road narrows on the right
Probability
0
10
20
30
40
0.0
0.1
0.2
0.3
0.4
0.6
0.8
1.0

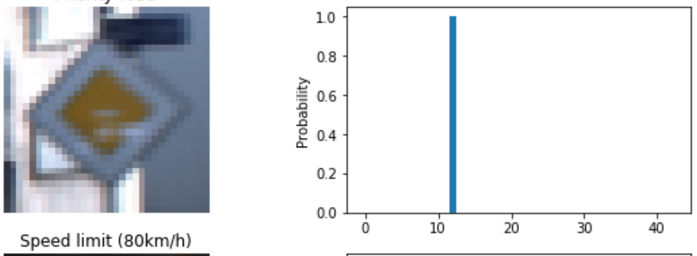
General caution



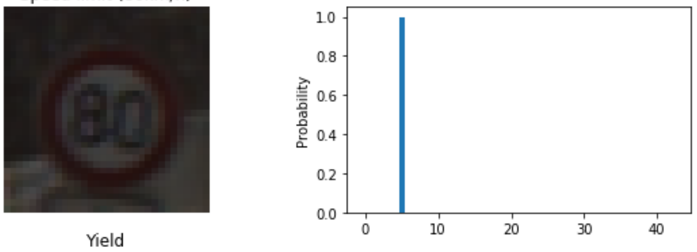
No passing for vehicles over 3.5 metric tons



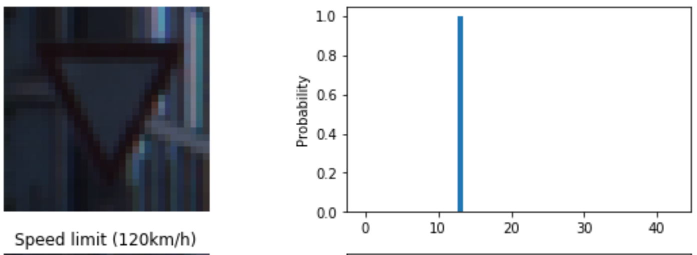
Priority road



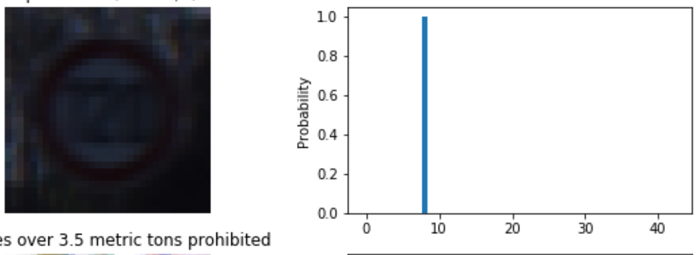
Speed limit (80km/h)



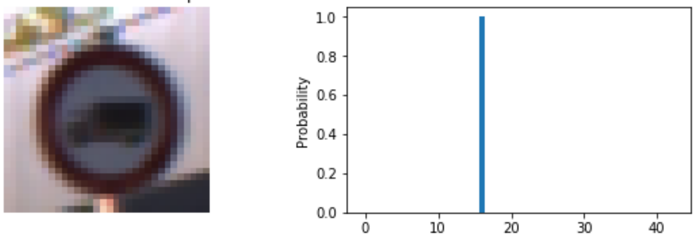
Yield



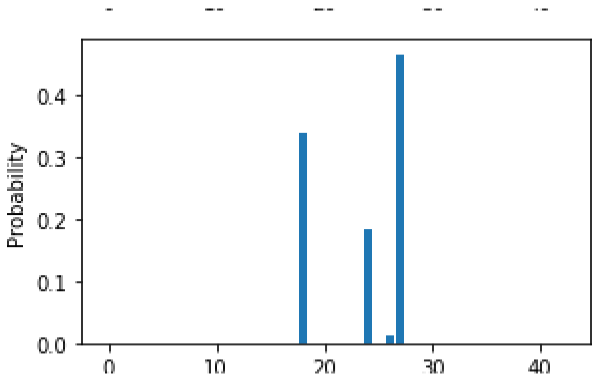
Speed limit (120km/h)



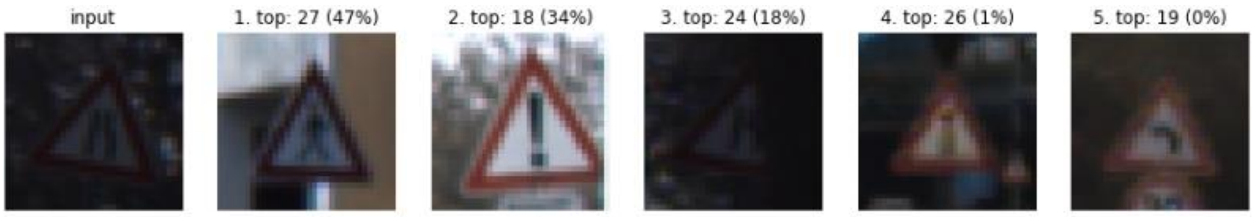
Vehicles over 3.5 metric tons prohibited



Let’s have a look to the performance of the sign “Road narrows to the right”:



Here the according top 5 results:



Here we can recognize that the first of top is the pedestrians-sign – a wrong recognition (47%), the second top the sign for general caution (34%) – also a wrong recognition. The 3rd top is a right recognition (18%). These signs are similar and the algorithm has difficulties to distinguish them.

2. Discuss the model's predictions on these new traffic signs and compare the results to predicting on the test set. Identify where in your code predictions were made. 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).

These aspects are discussed before also to give an answer on qualities and quantities of the difficulty of classifying signs.

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 and identify where in your code softmax probabilities were outputted. 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)

As we had assumed before there is only one of the 30 samples which prediction is under the probability of 0.6. All other sample signs are quiet good predicted.

Solution of Review Requirements

Files Submitted

Here is a link to my 1st Version of [project code](https://github.com/SDCND/CarND-Traffic-Sign-Classifier-Project/blob/master/Traffic_Sign_Classifier.ipynb) (included hyperlink: https://github.com/SDCND/CarND-Traffic-Sign-Classifier-Project/blob/master/Traffic_Sign_Classifier.ipynb

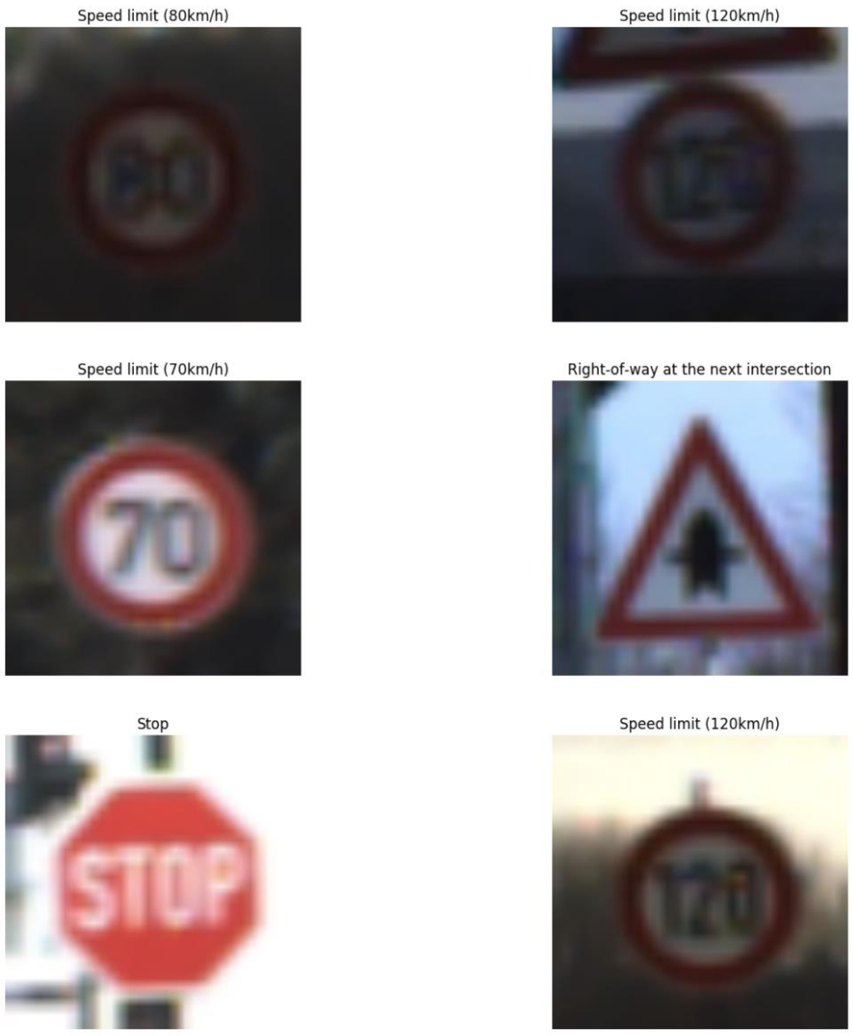
Here is the link of my 2nd Version of [project code](https://github.com/SDCND/CarND-Traffic-Sign-Classifier-Project/blob/master/Traffic_Sign_Classifier2.ipynb) (included hyperlink: https://github.com/SDCND/CarND-Traffic-Sign-Classifier-Project/blob/master/Traffic_Sign_Classifier2.ipynb

The solution of review requirements refers to the 2nd version of project code.

Files Submitted

The image quality (brightness, contrast, etc.) in the 1st version of project code version makes the prediction difficult. Therefore in the 2nd version of project code there is included an added method equalize() to prepare the images. After applying the equalizing mechanism the image quality is better and causes better results.

Sample images before equalizing:



Sample images after equalizing:

Test a Model on New Images

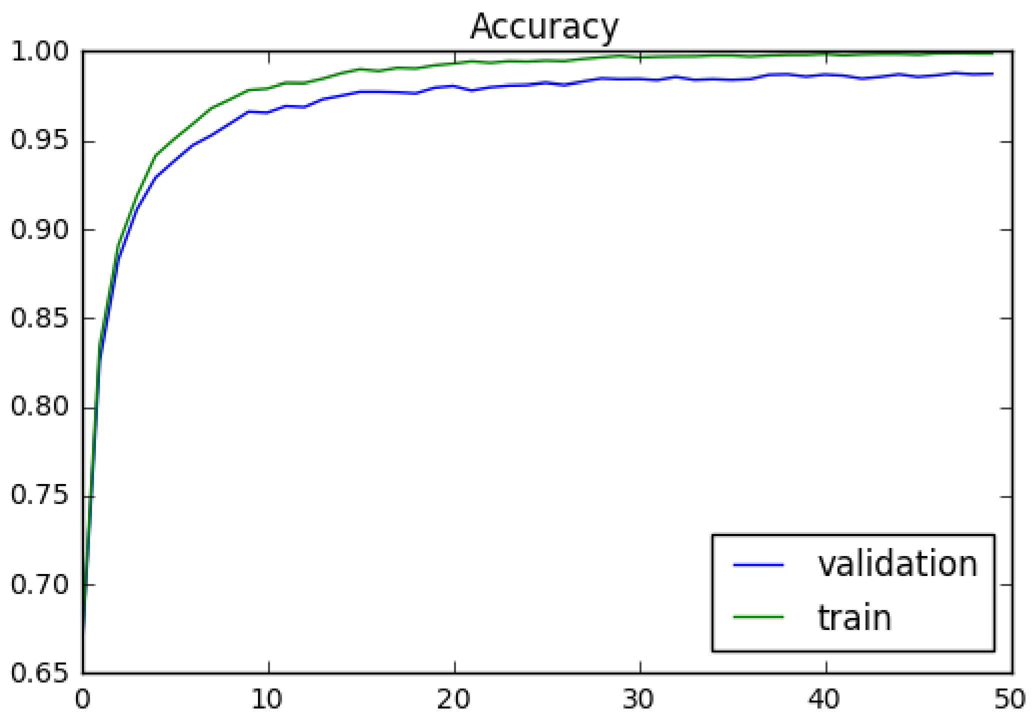
For the 2nd version of project code the print out of the final results after training are:

```
EPOCH 50 ...
Validation Accuracy = 0.987
Train Accuracy = 0.999

Model saved

Test Accuracy = 0.909
```

For the 2nd version of project code I have prepared a plot of the development of validation and training accuracies over the training epochs:



My final results were:

- training set accuracy of 0.999
- validation set accuracy of 0.987
- test set accuracy of 0.909

As complexity of a model increases, errors tend to get lower on training data. However, this doesn’t imply that testing error will also go down. When training error decreases without testing error also decreasing, this is called *overfitting*.

So the accuracy on the captured images is 99 % while it was 91 % on the testing set thus it seems the model is overfitting.