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.

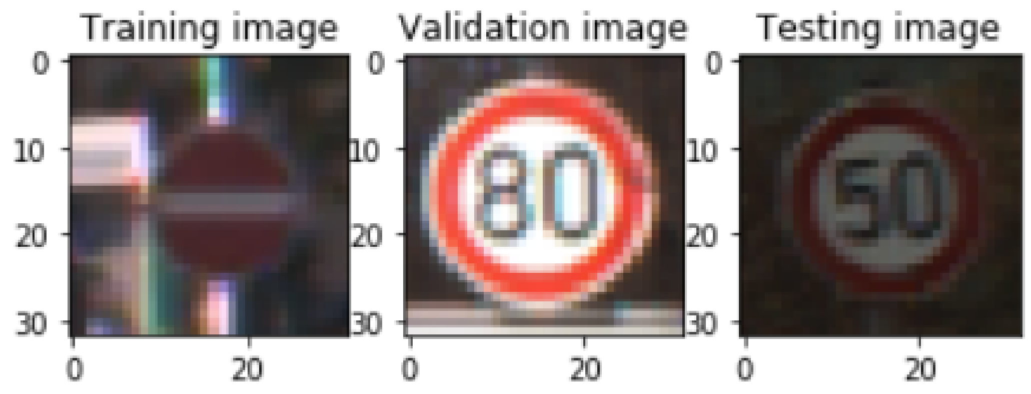
I used the pandas library to calculate summary statistics of the traffic signs data set:

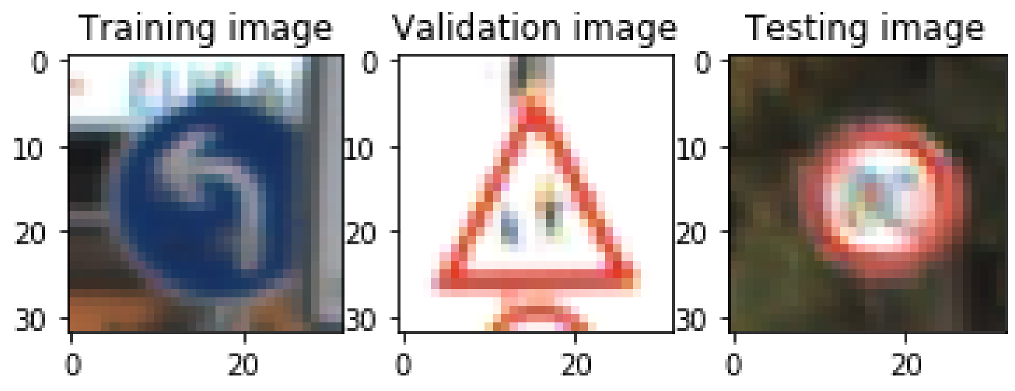
- The size of training set is ***34799***
- The size of test set is ***12630***
- The shape of a traffic sign image is ***32x32***
- The number of unique classes/labels in the data set is ***43***

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

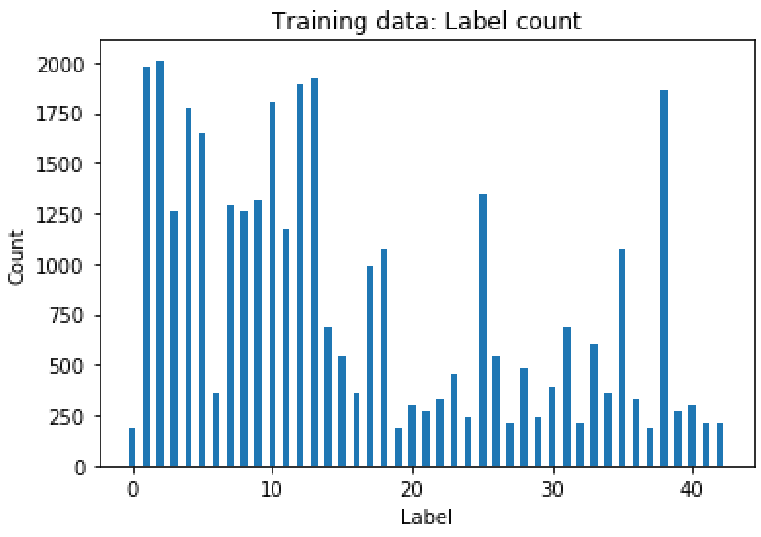
The code for this step is contained in the third code cell of the IPython notebook.

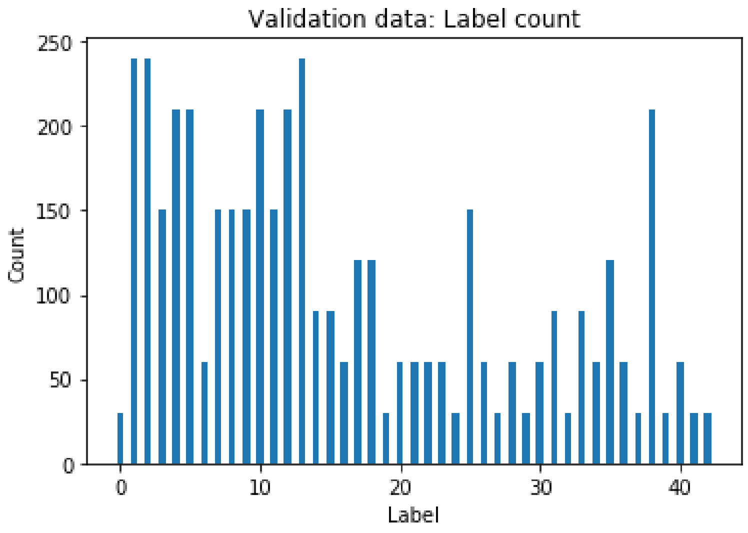
Here is an exploratory visualization of the (image) data set.

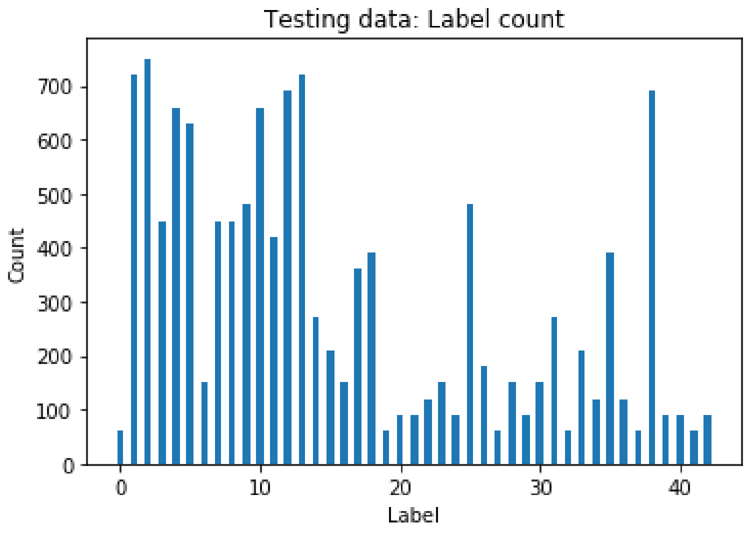




Here is a bar chart showing how the count of labels in each data set.







Design and Test a Model Architecture

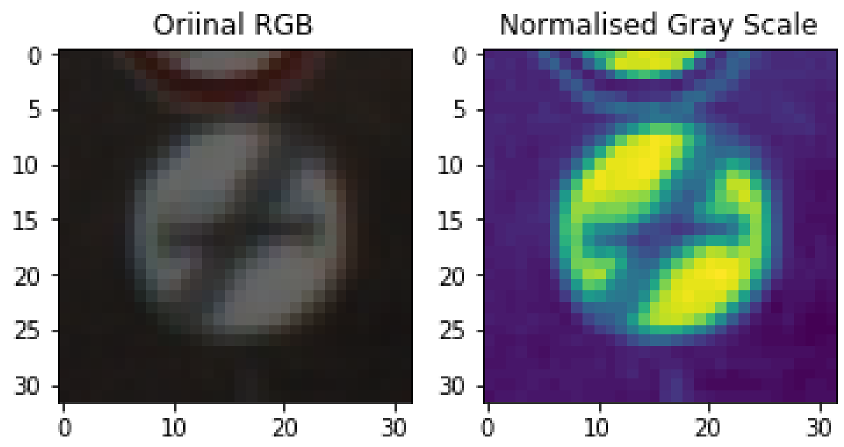
1. Describe how, and identify where in your code, you preprocessed the image data. What tecniques were chosen and why did you choose these techniques?

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

As a first step, I decided to convert the images to grayscale because

color does not provide the direct indication to the type of traffic signs. Same color is also use other signs, so the classification does not yield a lot of information on this. Secondly, we also reduce the size of data to speed up the process.

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



As a last step, I normalized the image data because …

it helps the gradient decent or optimizers to find the optimal solutions/weights. With the normalization, the scope of the input space is limited to a small region e.g. -1.0 to 1.0, in comparison with 0-255 range. Also the issue with the numerical stability of adding a big number with many small numbers over times (error accumulation).

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.

I merge the train and validation data together before using sklearn for sampling and splitting 60%-40% of training and validating data set.

X_train, X_valid, y_train, y_valid = train_test_split(X_train, y_train, test_size=0.4, random_state=42)

Note: Confirm the data is good (all labels in both data set and no bias) after splitting.

My final training set had 23783 number of images.

My validation set and test set had 15856 and 12630 number of images.

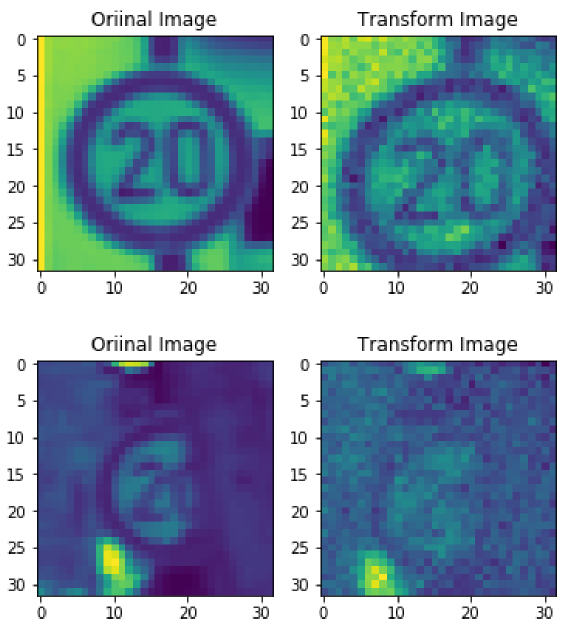
To cross validate my model, I randomly split the training data into a training set and validation set. I did this in *the fifth cell (in def Train) by using the validation set of data to check the accuracy of the training model obtained from the training data on every Epoch training loop.*

(Augment Optional)

The fourth code cell of the IPython notebook contains the code for augmenting the data set *(see Augment_ImageData)*. I decided to generate additional data because *it helps to generalize the model further by introducing the same sign with different characteristic/sense/pattern.*

To add more data to the the data set, I used the following techniques because *it changes a perspective on the image view and also noise features that may happen in the real applications.*

Here is an example of an original image and an augmented image:



The difference between the original data set and the augmented data set is the following ...

viewing perspective and additional random noises

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 fifth cell* of the ipython notebook.

My final model consisted of the following layers:

Layer	**Description**
Input	32x32x1 Grayscale
Convolution/1 (kernel: 5,5,1,6)	1x1 stride, valid padding, outputs 28x28x64
Relu /1	
Max pooling/1	2x2 stride, valid padding, outputs 14x14x6
Convolution/2 (kernel:5,5,6,16)	1x1 stride, valid padding, outputs 10x10x16
Relu/2	
Max pooling/2	2x2 stride, valid padding, outputs 5x5x16
Fully connected/3	Flatten, Linear, outputs 1x120
Relu/3	
Fully connected/4	Flatten, Linear, outputs 1x84
Relu/4	
Fully connected/5	Flatten, Linear, outputs 1x43
Logits/Softmax	Output 1x43

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.

To train the model, I used an ....

EPOCHS = 15

BATCH_SIZE = 400

RATE = 0.001

Having tuned various parameters, I found this hyperparameters have given the best performance.

The validation accuracy start to drop/settle at around EPOCHS 15.

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 fifth cell* of the Ipython notebook. *The accuracy is calculated in the 'Evaluate' function in 'CovNetwork' Class*

My final model results were:

- training set accuracy of ***95.5%***
- validation set accuracy of ***93.6%***
- test set accuracy of ***86.7%***

The steps to tune the hyper-parameters were

- fix learning rate to 0.001

- set batch size to 300

- tryout Epochs 30, 20- see the validation accuracy start to drop/settle at 15 Epochs

- vary batch size between 200-400 and check accuracy on the test data. Aiming to get the highest score avoid over-fitting the model by checking the accuracy on Train/Validation data

If a well known architecture was chosen:

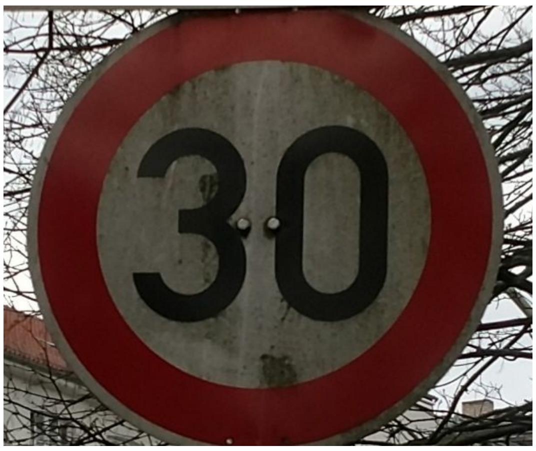
- **What architecture was chosen?**
  LeNet
- **Why did you believe it would be relevant to the traffic sign application?**
  The traffic signs have distinctive features such as triangle, circular, person icons, numbers, arrows that may happen in different layers. So I believe, the multi-layer of LeNet would be enough to capture these features on the different layers for the activation/prediction. Original architecture on these data set is justified with the good accuracy on both training, validating and also test data.
- **How does the final model's accuracy on the training, validation and test set provide evidence that the model is working well?**

  The accuracy from the training/validating data set was good and does not over fit, as well as the test accuracy is sufficient for the prediction (above 80%)

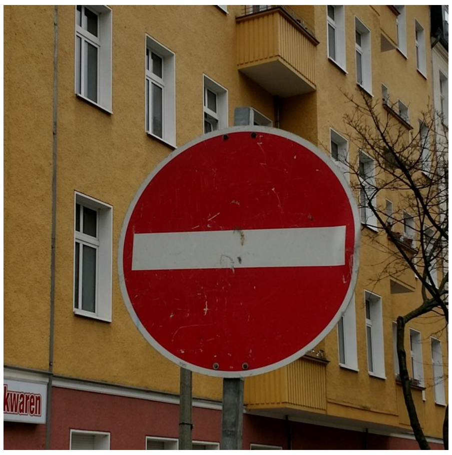
Test a Model on New Images

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.

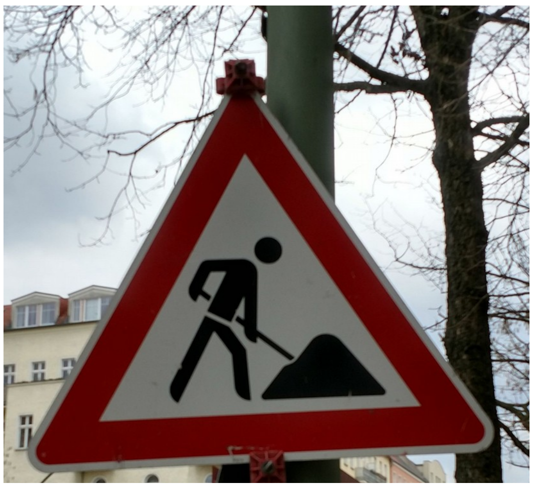
Here are five German traffic signs that I found on the web:



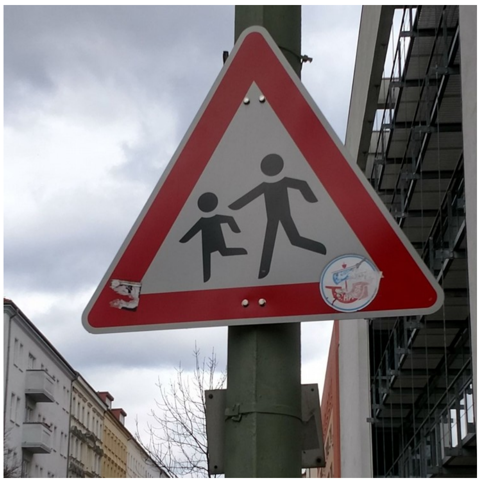
1.



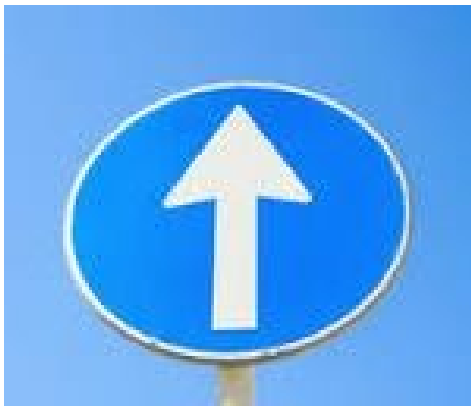
2.



3.



4.



5.

The first image might be difficult to classify because the dirty area (white) might create additional features on the image aw well as missing some curves at the top and bottom area.

The second image is not so bad for the classification apart from some additional features on the background.

The third image might be difficult to classify because of the background with many more features

The fourth image might be difficult to classify because there are some marks on the sign that obscure/distort some features on the sign. Actual test (below) on this image give the wrong prediction as the result.

The fifth image is not too difficult for the classification apart from the image quality/resolution.

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).

The code for making predictions on my final model is located in *the eighth&ninth* cell of the Ipython notebook. *(see GetPredictionUndSoftmax in the fifth cell, CovNetwork class)*

Image	**Prediction**
1/30 km/h speed limit	1/30 km/h speed limit
17/No entry	17/No entry
25/Road work	25/Road work
28/Children crossing	11/Right of way at the next intersection
35/Ahead only	35/Ahead only

The model was able to correctly guess 4 of the 5 traffic signs, which gives an accuracy of 80%. This is lower than the accuracy on the test set of 87.7 %. This is due to the limited resolution of the test sample of only 5 pictures than has the resolution of 20%. However it is comparable to the test set accuracy of 87.7%.

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)

The code for making predictions on my final model is located in *the 9 and 10 th cell* of the Ipython notebook.

For the first image, the model is quite sure that this is a No Entry sign (probability of 99.99 %). The top five soft max probabilities were

Probability	Prediction
9.99994516e-01	17/No entry
5.43349279e-06	14/Stop
3.04968481e-11	1/30 speed limit
1.52148329e-11	0/20 speed limit
3.00263156e-12	38/Keep right

For the second image, the model is quite sure that this is a Road work sign (probability of 99.99%). The top five soft max probabilities were

Probability	Prediction
9.99978542e-01	25/Road work
1.84194232e-05	22/Bumpy road
2.71555427e-06	29/Bicycle crossing
1.68717690e-07	24/Road narrow on the right
8.84061748e-08	31/Wild animal crossing

For the third image, the model is quite sure that this is a 30 speed limit sign (probability of 85.80%). The top five soft max probabilities were

Probability	Prediction
8.58015835e-01	1/30 speed limit
9.69238952e-02	17/No entry
4.38451655e-02	0/20 speed limit
6.37959805e-04	2/50 speed limit
2.52618367e-04	6/End of speed limit

For the fourth image (WRONG DETECTION), the model gives the wrong prediction on a Children crossing sign (probability of 0.064%). This is due to some marks on the sign surface that confuse the classifier. However, the correct prediction comes third in the list. The top five soft max probabilities were0

Probability	Prediction
9.63382185e-01	11/Right of way at the next intersection
3.57749127e-02	12/No passing for vehicle > 3.5 ton
6.42373809e-04	28/Children crossing
1.54211026e-04	30/Beware of ice, snow
4.06423933e-05	26/Traffic signals

For the fifth image, the model is very sure that this is a Ahead only sign (probability of almost 100%). The top five soft max probabilities were

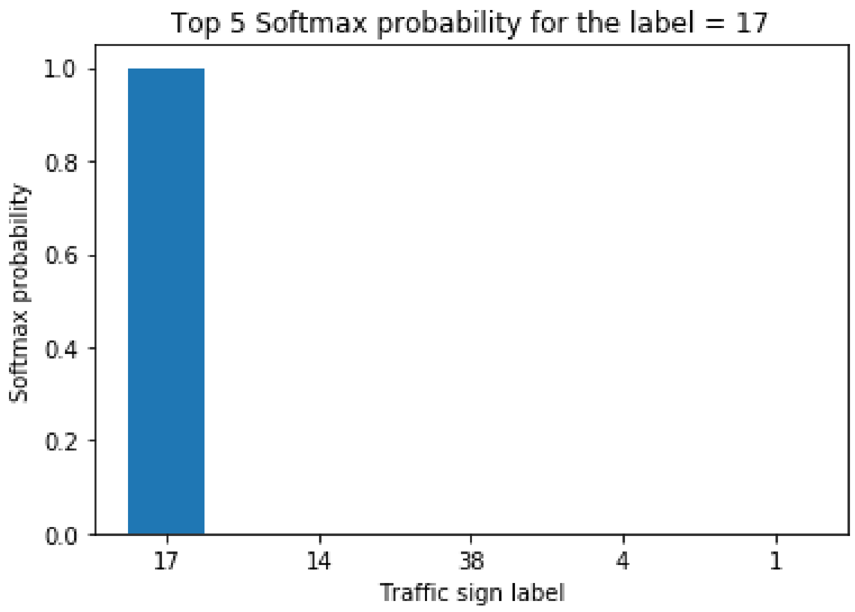
Probability	Prediction
1.00000000e+00	35/Ahead only
1.00596975e-09	9/No passing
6.47000342e-10	3/60 speed limit
6.35086669e-14	13/Yield
1.34218207e-14	36/Go straight or right

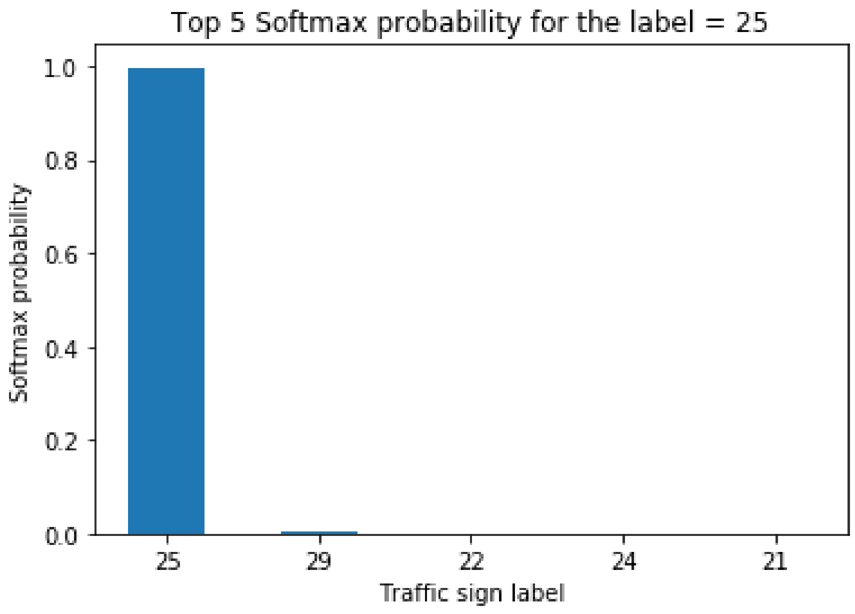
(Optional: Analyze New Image Performance in More Detail)

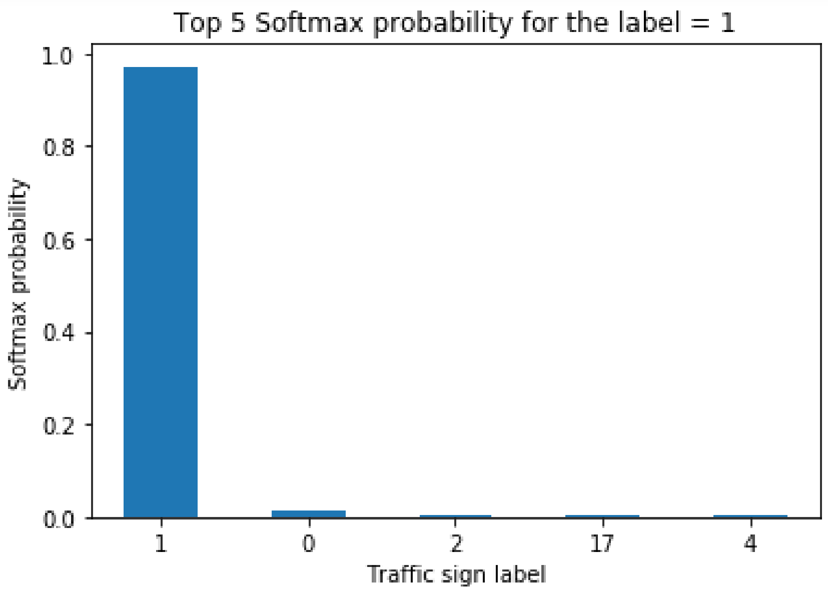
The code and bar chart plots are in the tenth cell. The plots provide the bigger of the accuracy on the trained model on each sign. More data or work could be done to improve on these signs to improve robustness.

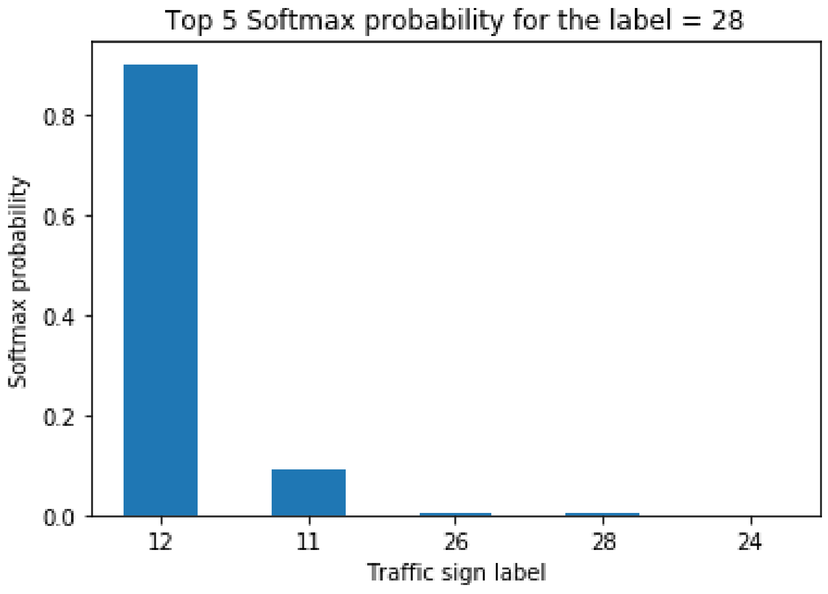
(Optional: Create Visualizations of the Softmax Probabilities)

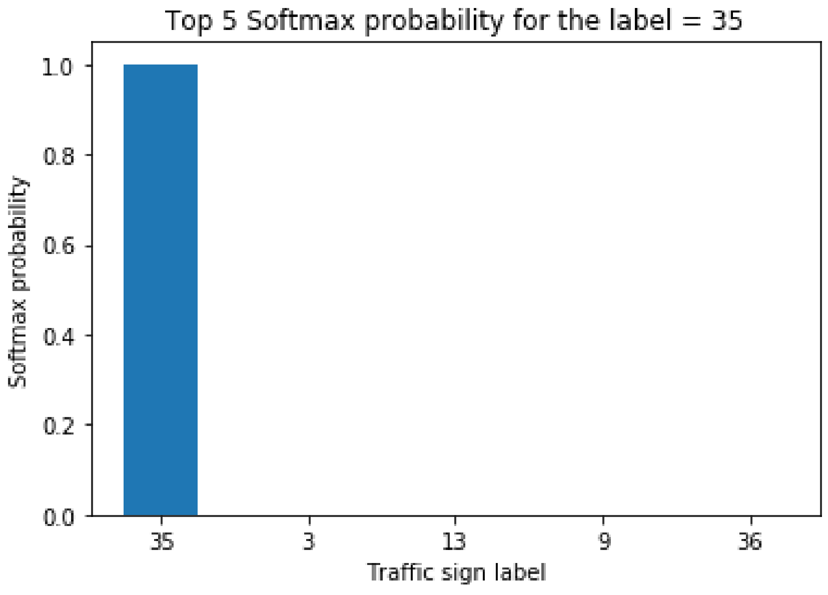
The code and bar chart plots are in the tenth cell.






Top 5 Softmax probability for the label = 1
Softmax probability
1.0
0.8
0.6
0.4
0.2
0.0
1
0
2
17
4
Traffic sign label


Top 5 Softmax probability for the label = 28
Softmax probability
0.8
0.6
0.4
0.2
0.0
12
11
26
28
24
Traffic sign label


Top 5 Softmax probability for the label = 35
Softmax probability
1.0
0.8
0.6
0.4
0.2
0.0
35
3
13
9
36
Traffic sign label