Traffic Sign Recognition Project Report

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

Here I will consider the rubric points individually and describe how I addressed each point in my implementation.

The code of my implementation please see Traffic_Sign_Classifier_.ipynb. and you will see the final results of my model here.

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.

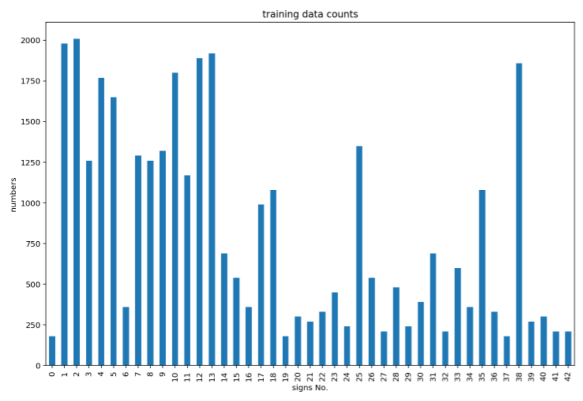
The code for **Data Set Summarized please see the second code cell** of My IPython Notebook, and the summary results of our data set please see table as below:

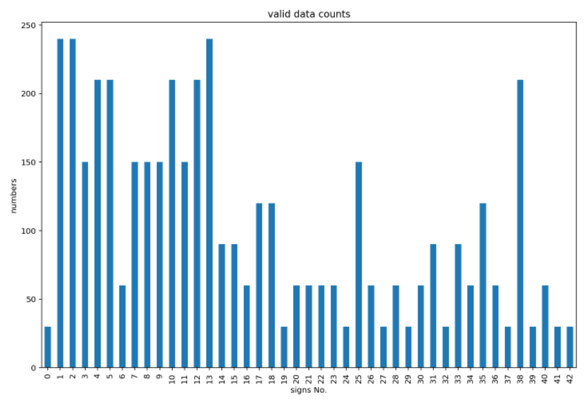
Number of training examples	34799
Number of testing examples	12630
Image data shape	(32, 32, 3)
Number of classes	43

Here I use pandas and numpy library to calculate summary statistics of the traffic signs data set.

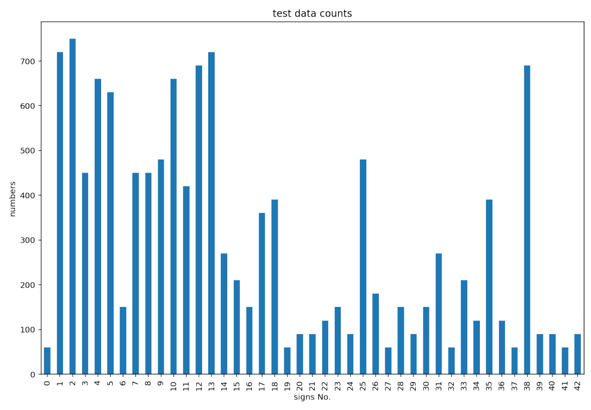
2. Include an exploratory visualization of the dataset and identify where the code is in your code file.

In addition, I use matplotlib to plot the bar chart to visualize the number of images of each classes. We can see for both training, valid and test data set, the number of each classes is unbalanced, and I also found the training and valid data set are all need randomized before we used them to train or valid our deep learning models. The dataset of training bar chart, valid bar chart and test bar chart please see as below:





Left:this is for training dataset;Right:this is for valid dataset



This is for test dataset

More details about the dataset visualization tasks, please see my Ipython Notebook(**from third code cell to eighth code cell**).

Table show as below is counts of training data set(before data augment process).

Sign No.	Sign Name	Sample Count	Sign No.	Sign Name	Sample Count
0	Speed limit (20km/h)	180	22	Bumpy road	330
1	Speed limit (30km/h)	1980	23	Slippery road	450
2	Speed limit (50km/h)	2010	24	Road narrows on the right	240
3	Speed limit (60km/h)	1260	25	Road work	1350
4	Speed limit (70km/h)	1770	26	Traffic signals	540
5	Speed limit (80km/h)	1650	27	Pedestrians	210
6	End of speed limit (80km/h)	360	28	Children crossing	480
7	Speed limit (100km/h)	1290	29	Bicycles crossing	240

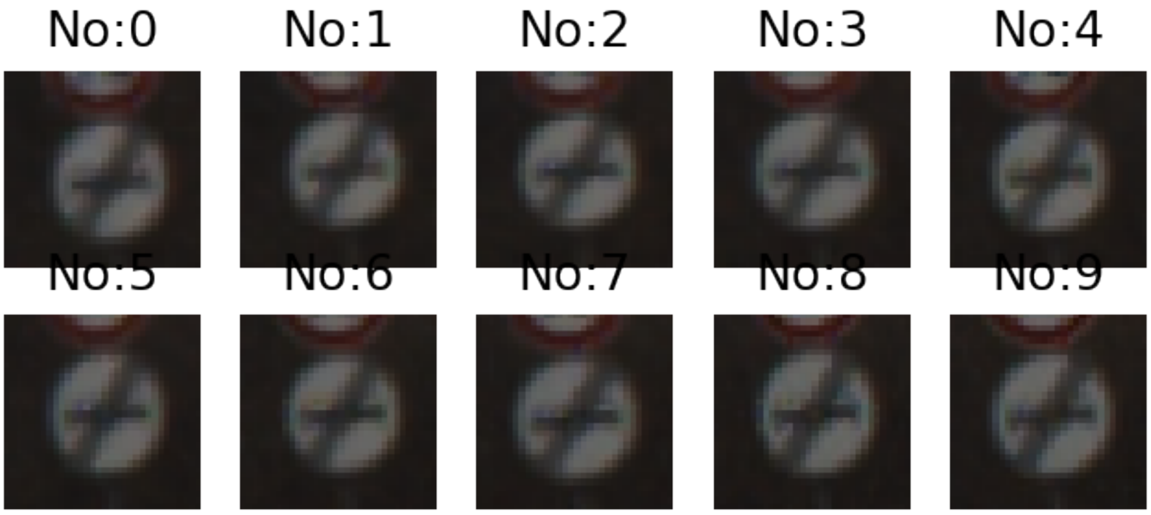
Sign No.	Sign Name	Sample Count	Sign No.	Sign Name	Sample Count
8	Speed limit (120km/h)	1260	30	Beware of ice/ snow	390
9	No passing	1320	31	Wild animals crossing	690
10	No passing for vehicles over 3.5 metric tons	1800	32	End of all speed and passing limits	210
11	Right-of-way at the next intersection	1170	33	Turn right ahead	599
12	Priority road	1890	34	Turn left ahead	360
13	Yield	1920	35	Ahead only	1080
14	Stop	690	36	Go straight or right	330
15	No vehicles	540	37	Go straight or left	180
16	Vehicles over 3.5 metric tons prohibited	360	38	Keep right	1860
17	No entry	990	39	Keep left	270
18	General caution	1080	40	Roundabout mandatory	300
19	Dangerous curve to the left	180	41	End of no passing	210
20	Dangerous curve to the right	300	42	End of no passing by vehicles over 3.5 metric tons	210
21	Double curve	270	Total	Traffic signals	34799

Design and Test a Model Architecture

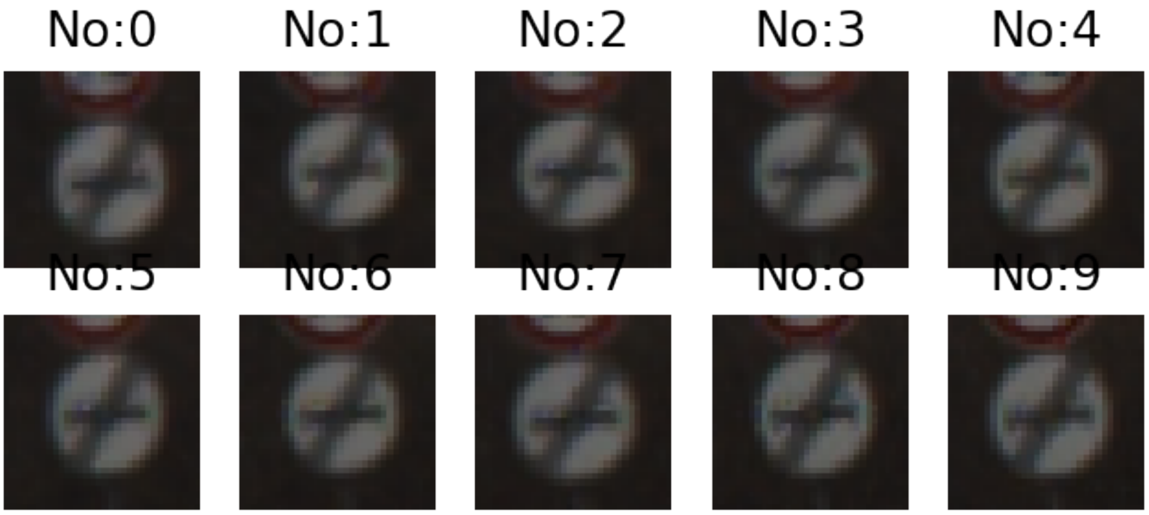
1. **Describe how, and identify where in your code, you preprocessed the image data. What tecniques were chosen and why did you choose these techniques? Consider including images showing the output of each preprocessing technique. Pre-processing refers to techniques such as converting to grayscale, normalization, etc.**

I convert the RGB image to Grayscale as input features for model training, but the results of the accuracy is not good than use RGB channel as input, I think the reason is for traffic signs classification task, the color shape is also very important. So in my final model, I remove the rgb2gray methods, and only normalize the data of each channel from 0 to 255 to 0 to 1.

The code for **image process with visualization parts please see the 9th code cell to 18th code cell.** and I only use image normalize technology to process the dataset for both train, valid and test data set. Also, we randomize the valid and test dataset here.



Before Normalize process



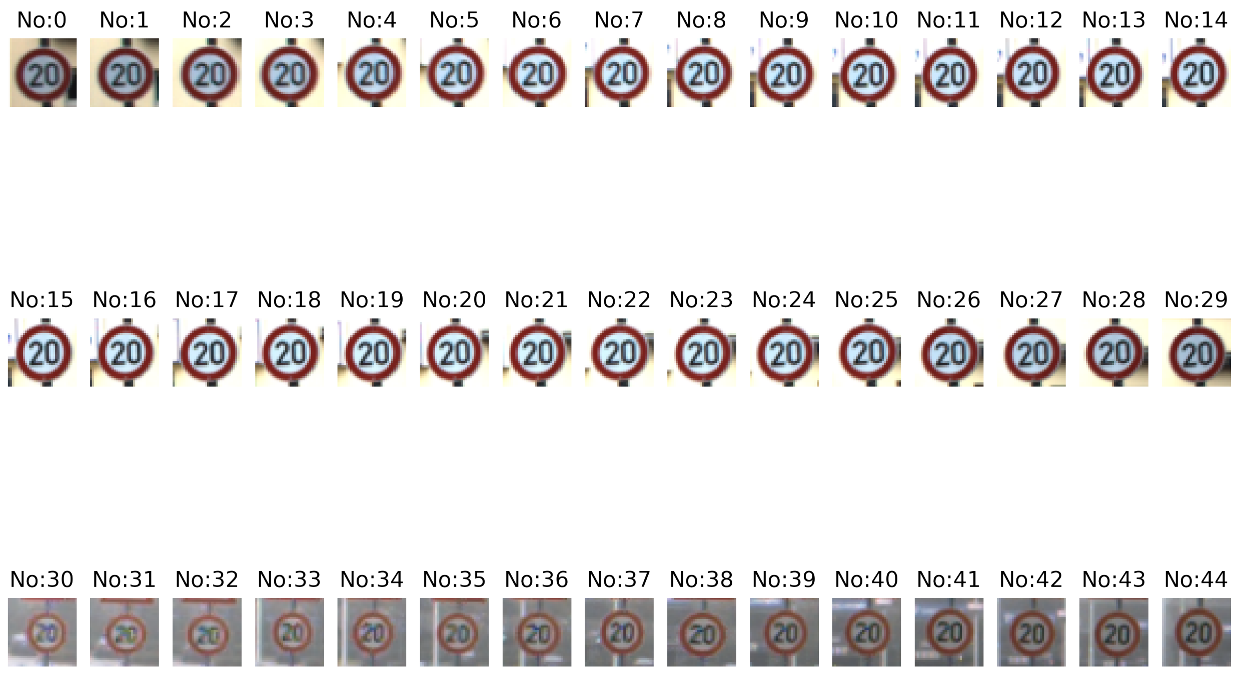
After Normalize process

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)

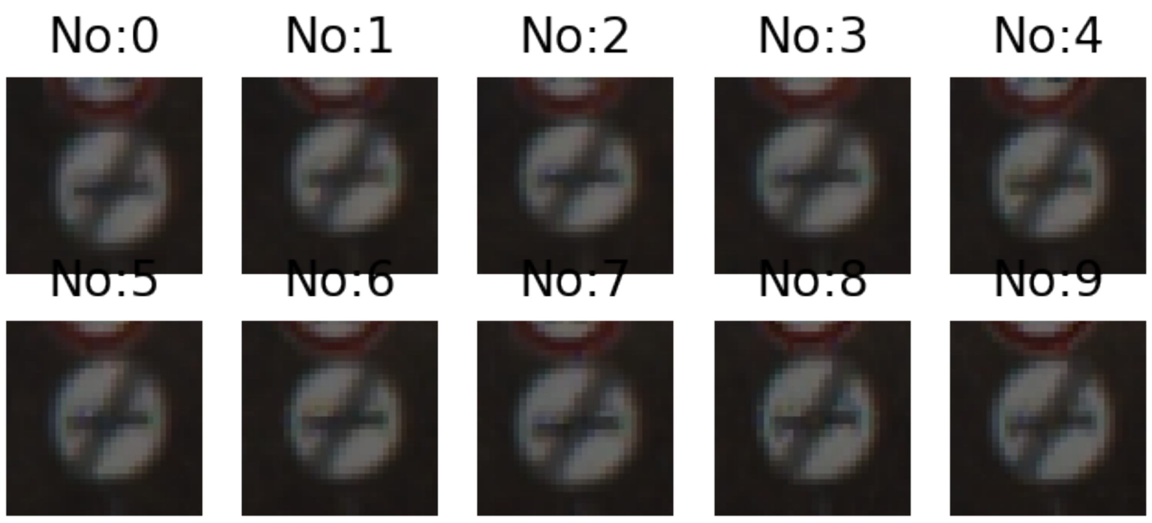
The code for **splitting the data into training and validation sets is contained in the first code cell** of the IPython notebook. We loaded training, validation and test dataset here. The training data was 34799(later we will augment the data to 5000 each classes), valid data was 4410 and test data was 12630.

As we discussed before, the training data set is unbalanced, using unbalanced dataset as training data to train our model will easily make our model overfitting. So I use some data augment technology to generate more data from original dataset and keep the number of each classes in same. We do not change the total number of valid and test data set(4410 for valid dataset, and 12630 for test dataset). However, the total number of training dataset are increased to 215000, and 5000 for each classes. Here we use brightness, contrast, saturation , gaussian noise and rotation technology to generate more data, the parameters of each methods are random generate, more details about them please see my Python Notebook from **19th code cell to 32 code cell**.

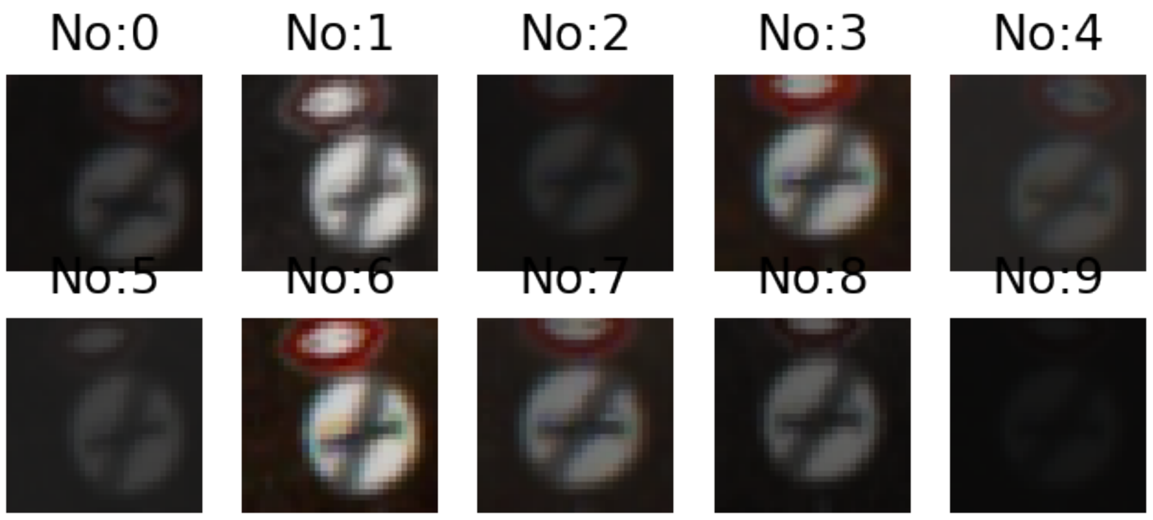
The image show as below is first 45 example visualize for our training dataset. The first 45 dataset is labeled as speed limit (20km/h).



Before we doing brightness, contrast, saturation and rotation technology. The origin images show as below(10 example images).



Then, we using random generate number to generate some parameters for both brightness, contrast, saturation and rotation operator. the results show as below.



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.

My final model consisted of the following layers: The **code is located in 33 to 47 code cell** of my Ipython Notebook

Layer	Description
Input	32 * 32 * 3 RGB image
Convolution	1x1 stride, same padding, 3 * 3 kernel size and 32 filters
Relu	activation function
Pooling	2x2 stride, 2 * 2 kernel size and use max_pooling
Dropout	keep_prob = 0.7
Convolution	1x1 stride, same padding, 3 * 3 kernel size and 64 filters
Relu	activation function
Pooling	2x2 stride, 2 * 2 kernel size and use max_pooling
Dropout	keep_prob = 0.7
Convolution	1x1 stride, same padding, 3 * 3 kernel size and 128 filters
Relu	activation function

Layer	Description
Pooling	2x2 stride, 2 * 2 kernel size and use max_pooling
Dropout	keep_prob = 0.7
Flatten	convert to 1D array
Fully connected with Relu	hidden node is 512
Dropout	keep_prob = 0.7
Fully connected with Relu	hidden node is 256
Dropout	keep_prob = 0.7
Output	output value to 43 classes
Softmax	Softmax operator

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.

To train the model , I applied optimizer methods is Adam, so I am not set the default learning rate here, and I set the batch size to 128, the number of epochs is 45, and the training loss of final epochs is 0.0556. The number of filters of each convolution layer, and the number of hidden nodes are see above(Table show above). The **code is located in 51 code cell of my** IPython Notebook.

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 48 and 50 code cell** of this Ipython Notebook. The **train parts is located in 52 code cell**.

My final model results were:

- training set loss is 0.0556
- validation set accuracy of 95.9%
- test set accuracy of 94.86%

I also develop an iterative approach here.

The first architecture I used for this task is LeNet, because I did not convert the image to grayscale, so I change the depth to other values and keep RGB channel as input depth to train the first model.

The problem of this initial architecture is the depth of convolution layer is too small, the model can not learn more edges or shapes of the 43 classes. So I change the number of output to other values and test the results with out sample data from website.

Here, I also trained the CNN by using residual block, and it takes more better performance than my final results, but it takes more computational power than my final methods, so I use the simple structure as my final results.

Also, because the overfitting problem will make our model have bad performance, I use dropout layer and batch normalization methods to avoid overfitting.

The code of my deep learning architecture please see the **34 code cell to 46 code cell of IPython Notebook.**

In CNN, each layer have forward and backward process, the flatten function, fully connected , activate function and output layer are similar in traditional artificial neural network architecture, and for both layers the forward operation and backward operation are all similar as ANN. we all use BP algorithm with gradient descent methods to train our model. However, in CNN we have convolution layer and polling layer. the convolution operation is very popular method used for signal process tasks(for example in canny edge detection algorithm, we use gaussian kernel as our convolution filter to blur the input image, then applied sobel operator as other convolution filter to output the 1st order gradient of x,y and gradient orientation. then to calculate the edge matrix of this image, finally use non-maximum suppression to product edge image). So we can train our convolution filter's kernel in backward process, also the pooling operation is upsample methods to reduce the computational. This is why the CNN can learning features by itself(don't like HoG!!).

Because of the overfitting problem will make model have very bad generalized skills, so normally we use batch normalization, dropout or L1/L2 Regularization methods to avoid it. here, I only use dropout methods.

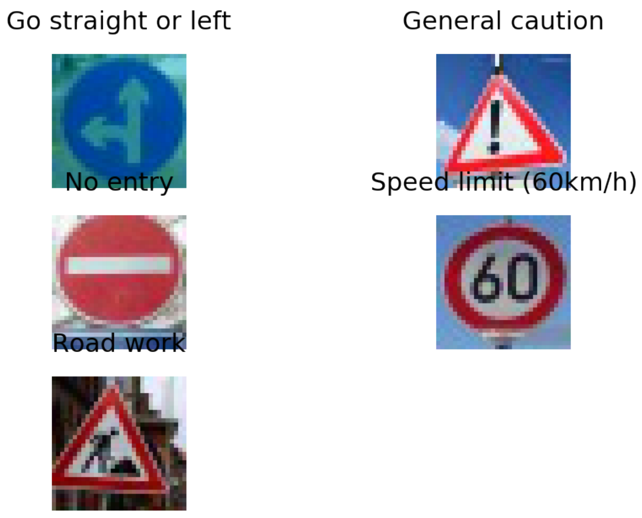
Test a Model on New Images

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:

Images	Label	Discuss
0	Go Straight or Left	The quality is good, and easy to classify
1	General Caution	Because the sign is rotation, so its a little difficult to classify
2	No Entry	The sign is big, little difficult to classify
3	Speen Limit(60 km/h)	easy
4	Road Work	quality is bad, difficult to classify

The image show as below



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).

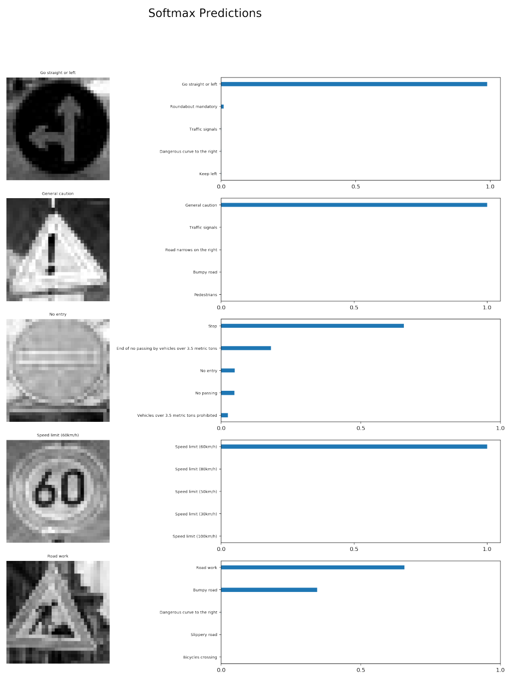
The code for making predictions on my final model is located in the **64 code cell of IPython Notebook.** and the results of final prediction is 80%.

Here are the results of the 5 signs prediction:

Image	Top 1	Top 2	Top 3	Top 4	Top 5	GroundTruth
0	Go Straight or Left	Roundabout mandatory	Traffic signals	Dangerous curve to the left	Keep left	Go Straight or Left
1	General Caution	Traffic signals	Road narrows on the right	Bumpy road	Pedestrians	General Caution
2	Stop	End of no passing by vehicles over 3.5 metric tons	No Entry	No Passing	Vehicles over 3.5 metric tons prohibited	No Entry
3	Speen Limit(60 km/h)	Speen Limit(80 km/h)	Speen Limit(50 km/h)	Speen Limit(30 km/h)	Speen Limit(100 km/h)	Speen Limit(60 km/h)
4	Road Work	Bumpy road	Dangerous curve to the right	Slippery road	Bicycles crossing	Road Work

Results Analyze:

We use test dataset and calculate the precision and recall for each 43 classes. we plot the top5 probability for results analyze here, the graphic visualization of the soft-max probabilities show as below.



For our test dataset, we have 94.86%, here we have 4 right predict out of 5 samples, our model are performs good at the first, second and fourth samples, and the final sample is nearly 60% at Road Work and nearly 40% at Bumpy Road. The third sample is wrong predict, the current answer only have less than 10% probability.

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 **67 code cell.**

The top five soft max probabilities can see the image before. and we also describe the final predict before. For the wrong prediction samples, may be the reason is color features , because the No Entry signs and Stop Signs are all have similar color features, our model may not learn well on those two signs. In addition, for the fifth sample, Road Work and Bumpy Road in low resolution ratio is very similar, so we have confused predict results.