Write up Template

You can use this file as a template for your write up 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 provided at the following link (https://review.udacity.com/#!/rubrics/481/view) individually and describe how I addressed each point in my implementation.

*****Write up/ 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 project code …
https://github.com/bnnair/SignalSignDetection-P3.git

*****Data Set Summary and 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.

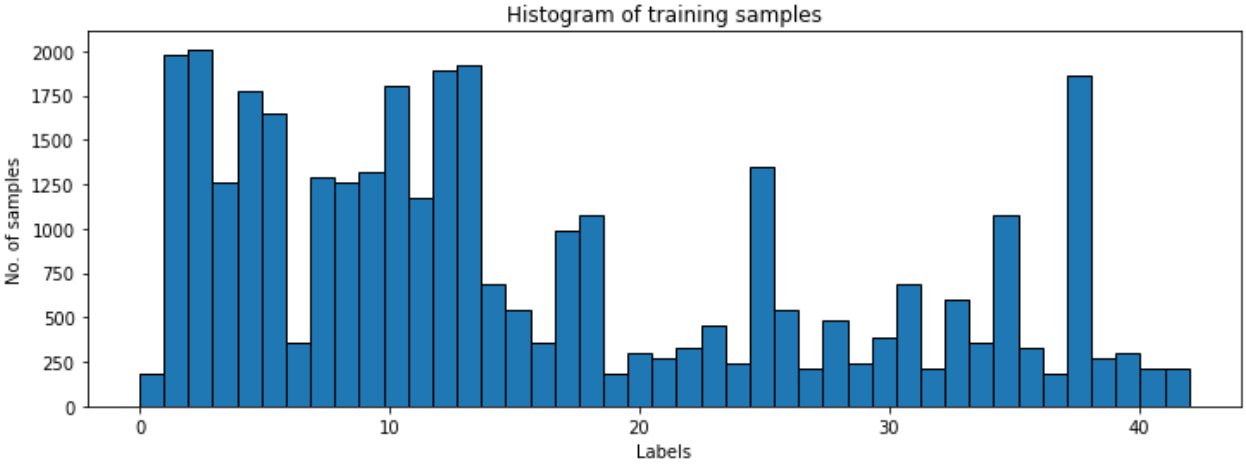
Since my local machine was getting hanged now and then, I did my project on an AWS EC2 instance created using the Deep Learning AMI Ubuntu server.
I uploaded the 3 pickle files for training, valid and testing dataset into S3.

I started with loading the 3 datasets using boto3 from S3.
Then I used pandas and numpy to get the 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

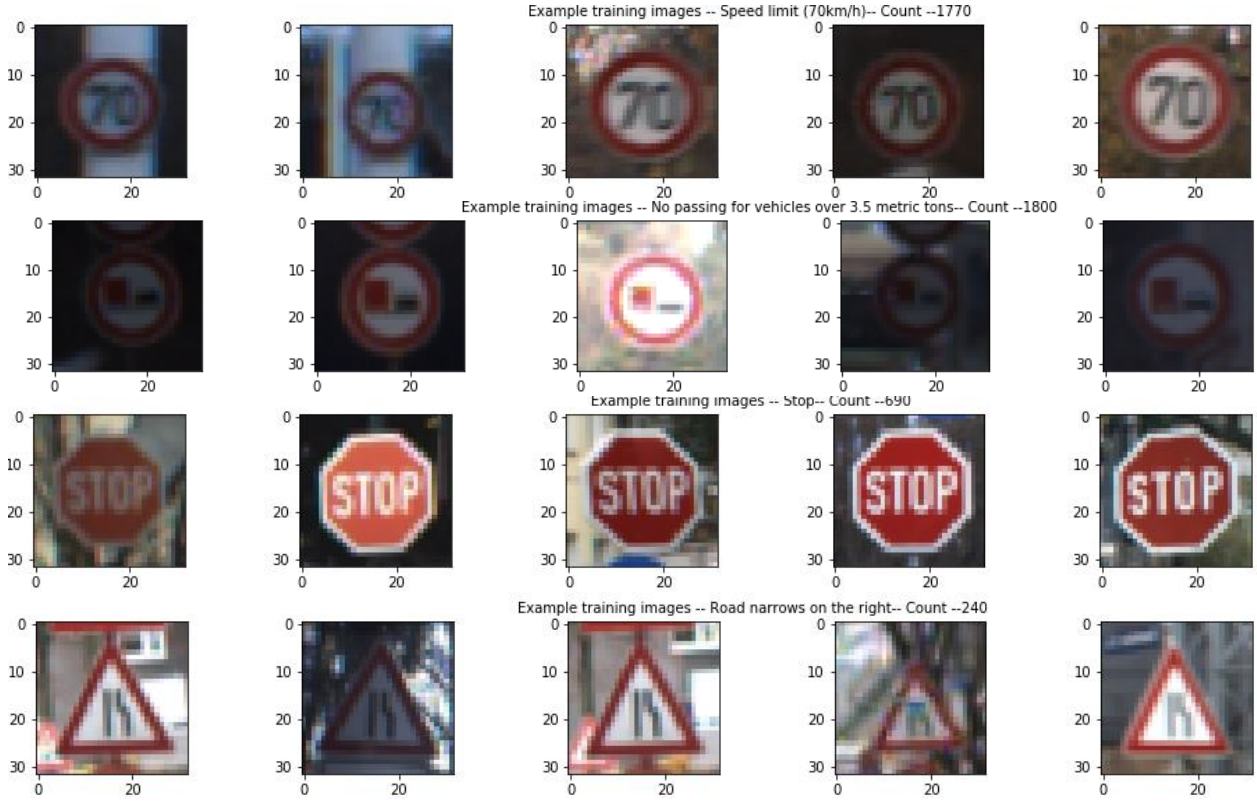
2. Include an exploratory visualization of the dataset

Here is the exploratory visualization of the training data set. It is a bar chart showing the count of data against each label.



This chart clearly shows that the distribution is not balanced across the classes. Apparently the dataset is unbalanced and some classes are represented significantly better than the others.

Let's now plot a bunch of random images for various classes to see what we are working with.



The images differ significantly in terms of contrast and brightness.
We would need to do some kind of transformation for the model to learn properly.

*****Design and Test a Model Architecture*****

1. Describe how you pre-processed the image data. What techniques were chosen and why did you choose these techniques? Consider including images showing the output of each pre-processing technique. Pre-processing refers to techniques such as converting to grayscale, normalization, etc. (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, and provide example images of the additional data. Then describe the characteristics of the augmented training set like number of images in the set, number of images for each class, etc.)

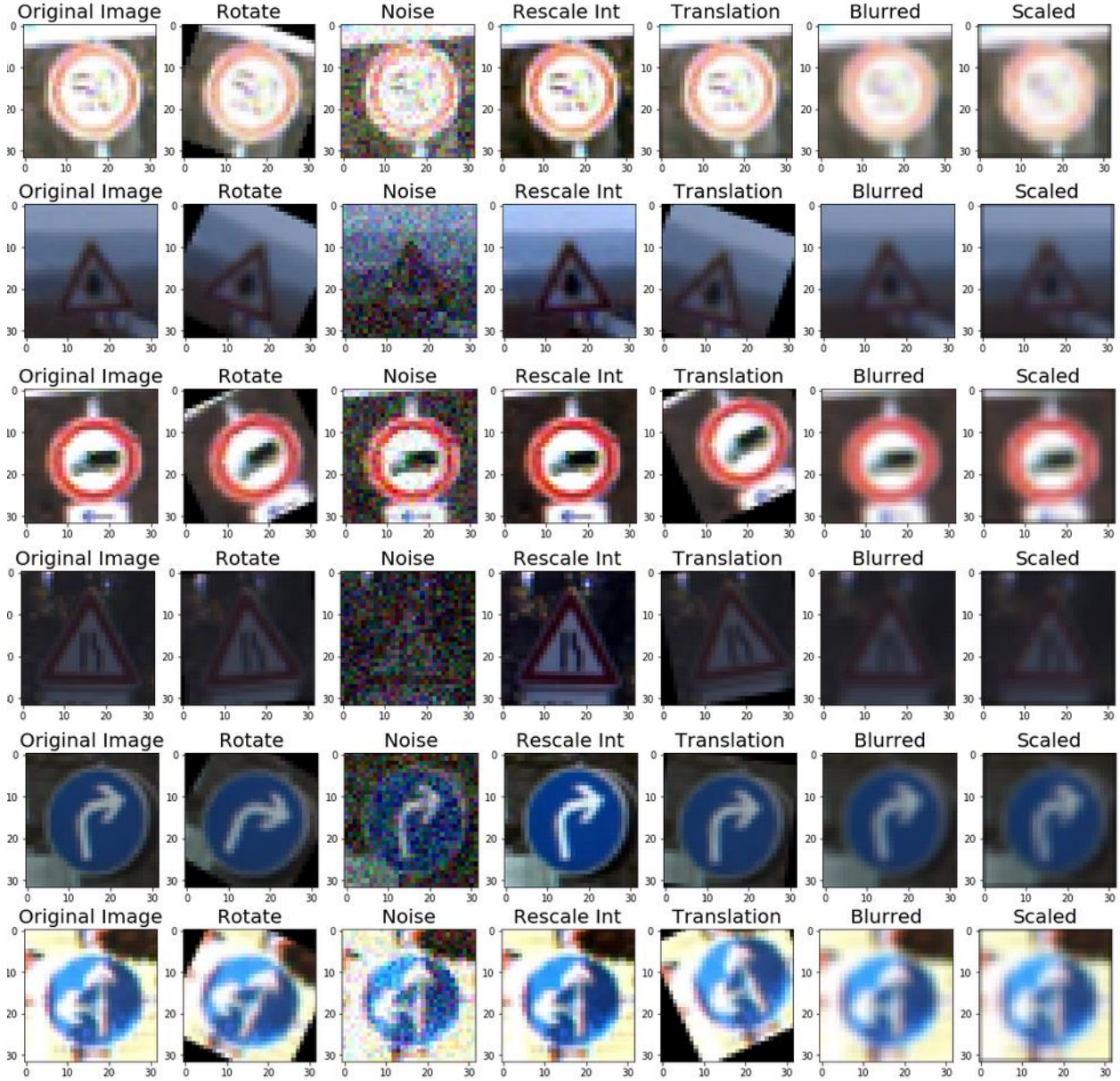
First since the dataset is unbalanced I decided to add few more images to the existing training data set to make it a balanced dataset.
I decided to add the difference of 4 times mean for each class to make the dataset balanced.

I went with the following data augmentation techniques:

a. Rotation : Rotation of the image at an angle
b. Gaussian noise: Introducing noise into the image
c. Rescale Intensity: Stretching and shrinking of the image intensity
d. Blurring: making a blurred image
e. Scaling: scaling out an image
f. Translation: horizontal translation of the image

For each class, random images were selected and transformed using a randomly selected one of the above functions. Thus the new images were added to the training data set.

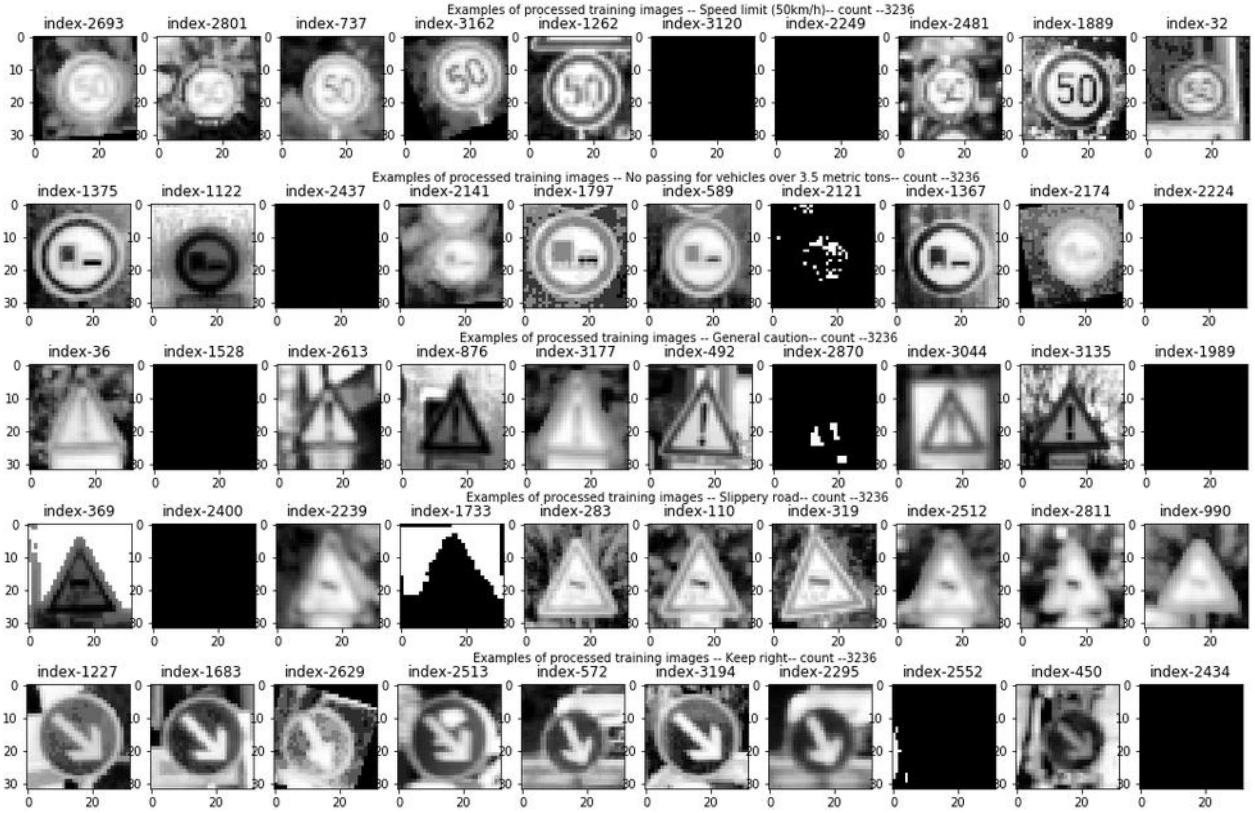
Few samples of the transformation can be seen below …



Once we got the balanced training dataset , the following operations were performed on the dataset..

a. Converting the images to grayscale. This is done in order to reduce the model computation. Also as per Yann LeCunn, using color channels did not seem to improve the model performance a lot.
b. Histogram Equalisation: Histogram Equalization is a method that adjusts image intensities in order to enhance the contrast of the image. It is straightforward to apply this function on a grayscale image as the method just equalizes the histogram of a grayscale image.
c. Normalize the image: Since the images color values range from 0 to 255, it is important to normalize the range to better model performance. I normalized it so that it ranged between -1 to +1

Here are images of sample data after the above transformation..



Even though some of the above images seem very dark , however if drawn to a slightly large scale individually, the darkness reduces.

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.

My final model consisted of the following layers:

Layers	Description
Input	32 x 32 x 1 (after transformation to grayscale)
Convolution Layer-1	Filter - 3 x 3 x 1 x 32, stride - 1 x 1 , padding - Valid, output - 30 x 30 x 32
Activation Layer	Relu
Average Pooling	input-30 x 30 x 32, output - 15 x 15 x 32
Convolution layer-2	Filter - 3 x 3 x 1 x 32, stride - 1 x 1 , padding - Valid, output - 13 x 13 x 64
Activation Layer	Relu
Average Pooling	input-13 x 13 x 64, output - 6 x 6 x 64
Convolution layer-3	Filter - 3 x 3 x 1 x 64, stride - 1 x 1 , padding - Valid, output - 4 x 4 x 128
Activation Layer	Relu
Average Pooling	input-4 x 4 x 128, output - 2 x 2 x 128
Flatten	output - 512
Fully Connected Layer-1	input - 512 , output - 256
Activation layer	Relu
Drop out layer	keep_prob - 0.5
Fully Connected Layer-2	input - 256 , output - 128
Activation layer	Relu
Drop out layer	keep_prob - 0.5
Fully Connected Layer-3	input - 128 , output - 43

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.

For optimizer I used AdamOptimizer. It is said to be better than the Gradient Descent Optimizer. I uses moving average of parameters and produces good results after few learning steps , while gradient descent would require much more number of learning steps.
Tried few numbers of epochs and finalized on 100.
I went with a batch size of 128 and number of epochs of 100.

For learning rate , first I choose a fixed learning rate of 0.003 , however, the accuracy was not crossing 92 as well as the optimizer was not converging.
So I then went with exponential decay learn rate and chose the following hyper parameters:
Starting learn rate = 0.003
Decay steps = 10000
Decay rate = 0.96

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:
* Validation set accuracy of: **97.3%**
* Test set accuracy of: **93.8%**

If an iterative approach was chosen:

*** What was the first architecture that was tried and why was it chosen?**
The architecture proposed is inspired by Yann LeCun's paper on classification of traffic signs. I have done some tweaks based on the performance on the model by varying filter sizes, depth and number of convolution layers, as well as dimensions of fully connected layers. I thought that this architecture from Yann LeCun would be a good starting point where they established a new record of 99.17% accuracy.

*** What were some problems with the initial architecture?**
Initially I went with 2 convnet layers and 2 fully connected layers with relu as activation and max pooling. Initially went with 32 filters for both the layers, but problem was that I was not able to run it on my local machine as it would throw memory error. Then I spawn a new AWS EC2 machine with large memory, which provided good performance and the model ran without any memory problem.

*** How was the architecture adjusted and why was it adjusted?**

Tried the above model with fixed learn rate of 0.003 , max pool size of 2x2 and drop out of 0.5 only in the fully connected layer. Different convnet filters size, from 5x5 to 3x3 were also tried.
One common justification for adjusting architecture would be due to overfitting or underfitting. A high accuracy on the training set but low accuracy on the validation set indicates over fitting; a low accuracy on both sets indicates under fitting.

*** Which parameters were tuned? How were they adjusted and why?**
The max pool replaced the avg pool which did not give much of a performance.
The drop out was introduced only in the fully connected layer as the drop out in the convnet layer was of not much use based on some research.
Exponential decay of learn rate was used so that the learn rate would decay as the optimizer approaches convergence.
The network is trained using mini-batch stochastic gradient descent with the Adam optimizer.

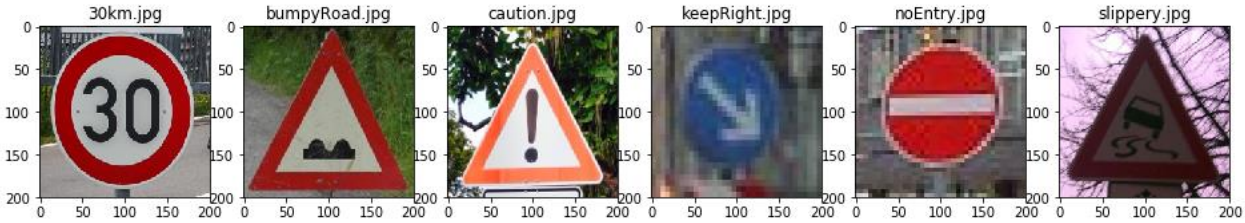
*** What are some of the important design choices and why were they chosen? For example, why might a convolution layer work well with this problem? How might a dropout layer help with creating a successful model?**
In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing and classifying visual imagery.
CNNs use a variation of multilayer perceptrons designed to require minimal preprocessing. CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage.

In order to improve the model reliability, dropout was introduced, which is a form of regularisation where weights are kept with a probability *p*: the unkept weights are thus “dropped”. This prevents the model from overfitting.

*****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 six German traffic signs that I found on the web:



All the signs were successfully classified correctly. If we run the prediction for a few times, there may be one or two images that are wrongly classified.
It's mostly the "Keep Right sign", since the image is not clear.

```
INFO:tensorflow:Restoring parameters from ./lenet
Predicted Labels
[ 1 22 18 38 17 23]

1 - Speed limit (30km/h) --> Correct
22 - Bumpy road --> Correct
18 - General caution --> Correct
38 - Keep right --> Correct
17 - No entry --> Correct
23 - Slippery road --> Correct
```

3. 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).

The model's prediction on these new traffic signs were 100% accurate , however there may be some times if we run the prediction for a few time, the accuracy falls as one or two signs are predicted in-accurately.
The test set accuracy is **93.8%,** which means that there are still some test images which are not predicted properly maybe because some of the images are very blurry, despite our histogram equalization, while others seem distorted.
We probably don't have enough examples of such images in our test set for our model's predictions to improve.

Image	Prediction
30km	30km
BumpyRoad	BumpyRoad
Caution	Caution
KeepRight	KeepRight
NoEntry	NoEntry
Slippery	Slippery

4. 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)

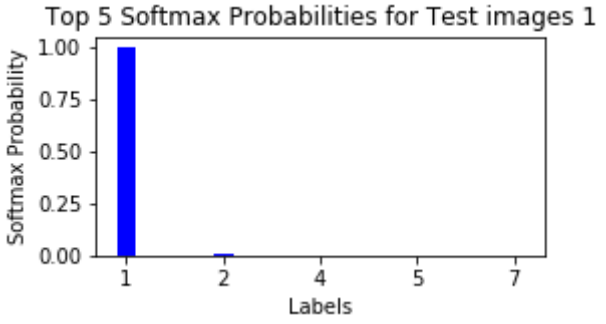
The code for making predictions on my final model is located in the second last cell of the Ipython notebook.

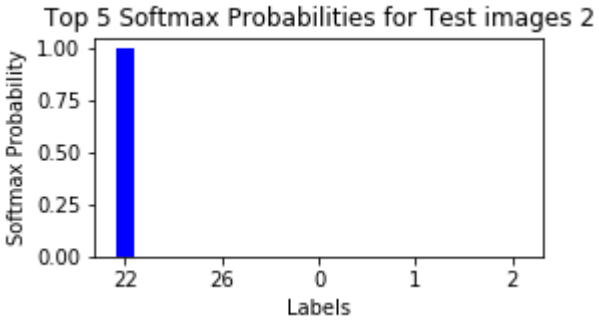
For the first image, the model is precisely sure that this is a 30km sign (probability of 0.99), and the image does contain a 30km sign. The top five soft max probabilities were as shown below…

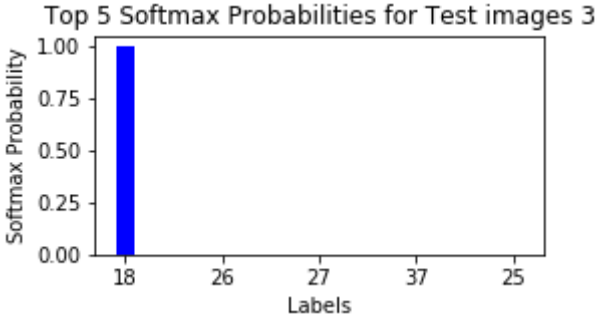
Prediction	probability
30km	0.995
BumpyRoad	1
Caution	1
KeepRight	1
NoEntry	1
Slippery	1

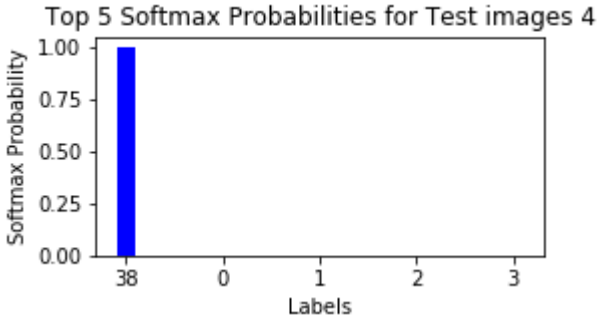
```
Top 5 softmax probabilities for each test image
TopKV2(values=array([[9.9514991e-01, 4.7418410e-03, 1.0255267e-04, 3.0383555e-06,
        1.6835462e-06],
       [1.0000000e+00, 2.5710465e-30, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00],
       [1.0000000e+00, 1.1668735e-10, 7.2813198e-25, 1.2053483e-28,
        1.5127654e-32],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00],
       [1.0000000e+00, 1.9150954e-14, 1.2639526e-18, 7.4157822e-19,
        4.9435389e-19]], dtype=float32), indices=array([[ 1,  2,  4,  5,  7],
       [22, 26,  0,  1,  2],
       [18, 26, 27, 37, 25],
       [38,  0,  1,  2,  3],
       [17,  0,  1,  2,  3],
       [23, 19, 28, 31, 20]], dtype=int32))
```

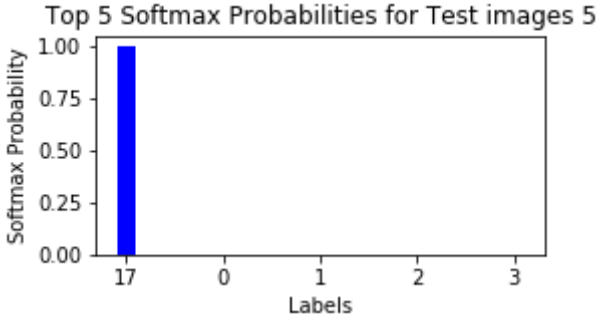
Visualization of the Softmax probabilities by a bar chart.

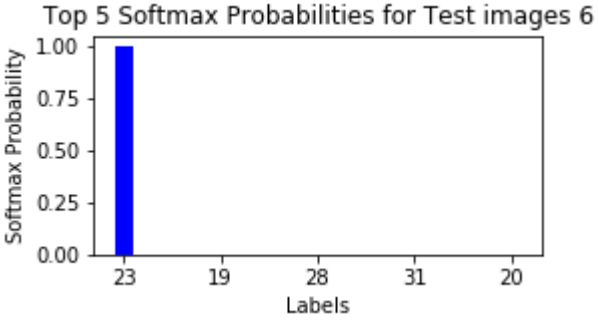












Stand Out Suggestions from Rubric

1. Data Augmentation.

Since our dataset was imbalanced as shown above, I decided to make the dataset more balanced by adding more images for the labels having less representation in the dataset.

I went with the following data augmentation techniques to add more images to the training data set:

a. Rotation : Rotation of the image at an angle
b. Gaussian noise: Introducing noise into the image
c. Rescale Intensity: Stretching and shrinking of the image intensity
d. Blurring: making a blurred image
e. Scaling: scaling out an image
f. Translation: horizontal translation of the 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?

Not been able to do the above visualization. Will be doing it soon and upload again.