Traffic Sign Classifier

1)Data Set Summary & Exploration

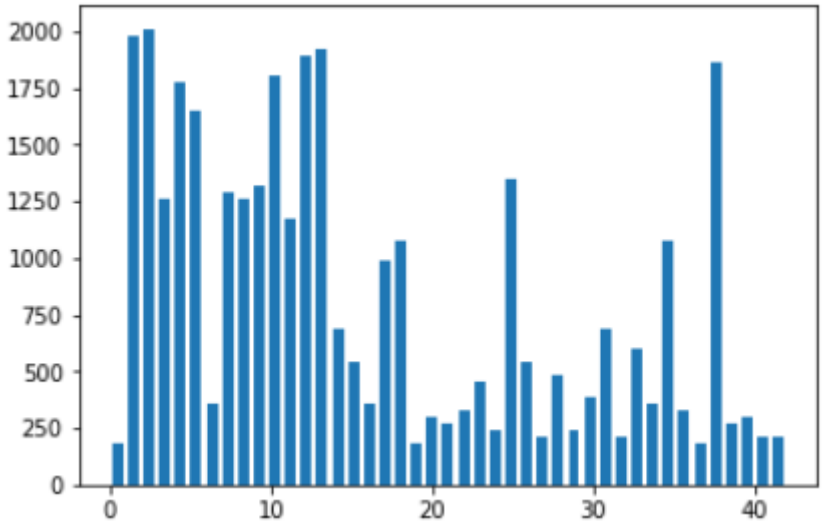
I used the numpy library to calculate summary statistics of the traffic signs data set:

- The size of training set is 34799
- The size of 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

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

Data was initially stored in pickle files in which are imported using the following code:

```
training_file = '../data/train.p'
validation_file='../data/valid.p'
testing_file = '../data/test.p'
```



Training Data Chart

2)Visualization of training data

Random set of ten images and labels are displayed to visualize the training data set.

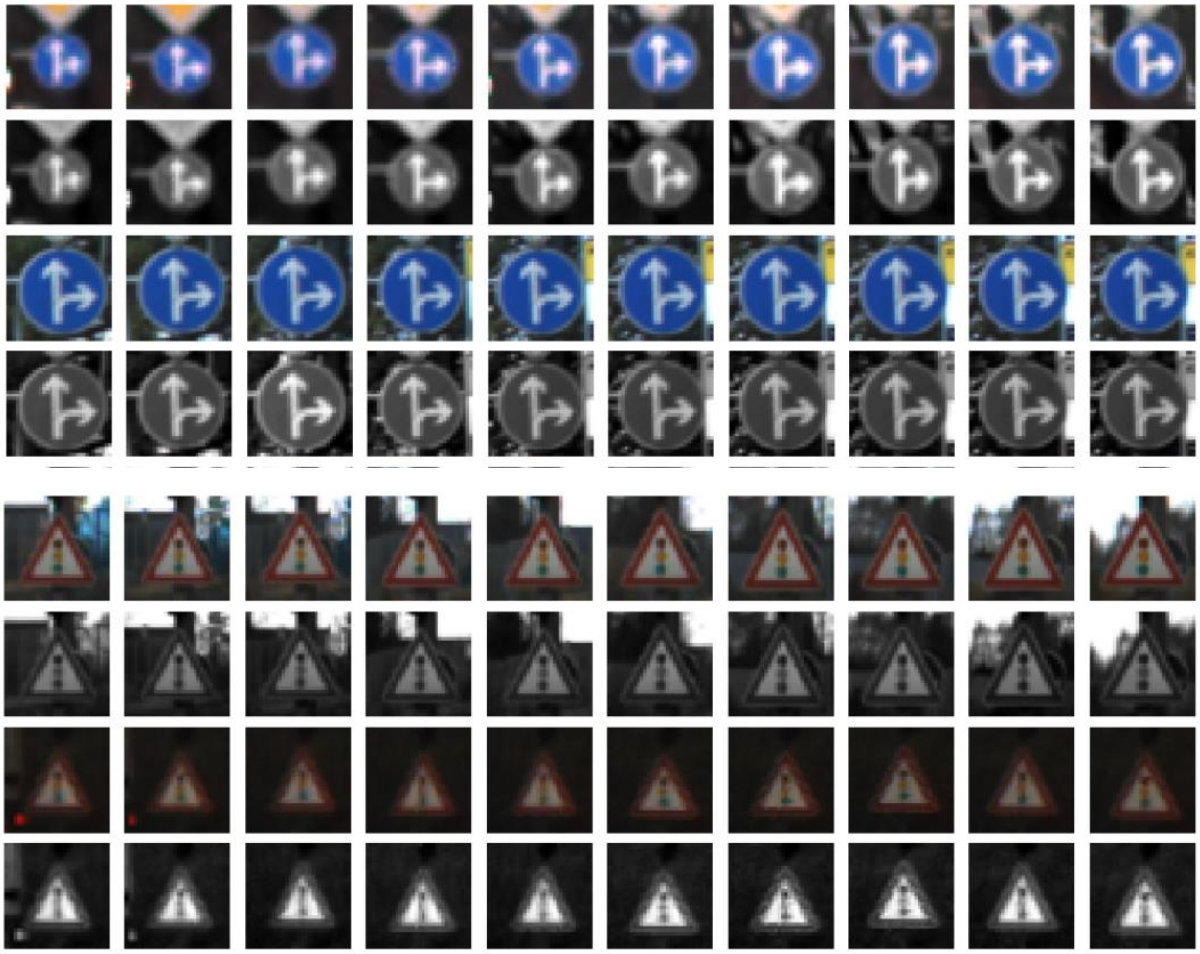
Training dataset images consists of traffic sign of variable brightness and variable values of colours shot in different angles.

3)Image Pre-processing:

I decided to convert the images to grayscale because in the technical paper it outlined several steps they used to achieve 99.7%.

This worked for me as the excess information only adds extra confusion into the learning process. After the grayscale I also normalized the image data because I've read it helps in speed of training and performance because of things like resources.

Here are some images on how grayscaling and normalization transformed my images:



4)Model Architecture

Below is the model architecture I have used which is a slight modification to the code given by Udacity Instructor's Code

1. 5x5 convolution (32x32x1 in, 28x28x6 out)
2. ReLU
3. 2x2 max pool (28x28x6 in, 14x14x6 out)
4. 5x5 convolution (14x14x6 in, 10x10x16 out)
5. ReLU
6. 2x2 max pool (10x10x16 in, 5x5x16 out)
7. 5x5 convolution (5x5x6 in, 1x1x400 out)
8. ReLu
9. Flatten layers from numbers 8 (1x1x400 -> 400) and 6 (5x5x16 -> 400)
10. Concatenate flattened layers to a single size-800 layer
11. Dropout layer
12. Fully connected layer (800 in, 120 out)
13. ReLu
14. Fully connected layer (120 in, 84 out)
15. ReLu
16. Output layer (84 in, 43 out)

5)Model Training & Solution Approach

For training of model I used the following value

```
EPOCHS = 60
BATCH_SIZE = 96
rate = 0.0009
```

These values gave me an accuracy of 0.935

Other than this I tried giving Epochs = 60 and Batch_Size = 128 which gave me an accuracy of ~0.92

Also the values of accuracy differed by a slight value and gave me the best output accuracy of 94.5

6)Testing model on new images

These are the images taken from the German Traffic Sign Dataset

7)Performance on new images

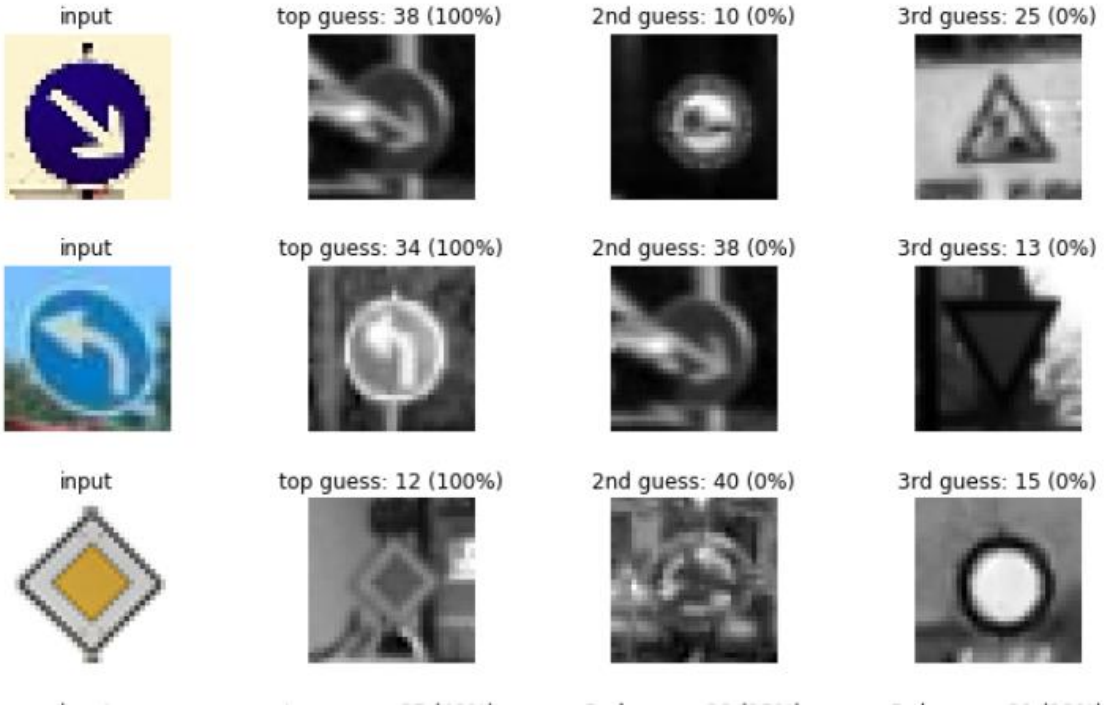
These images are first gray scaled and then normalized and passed through the CNN.

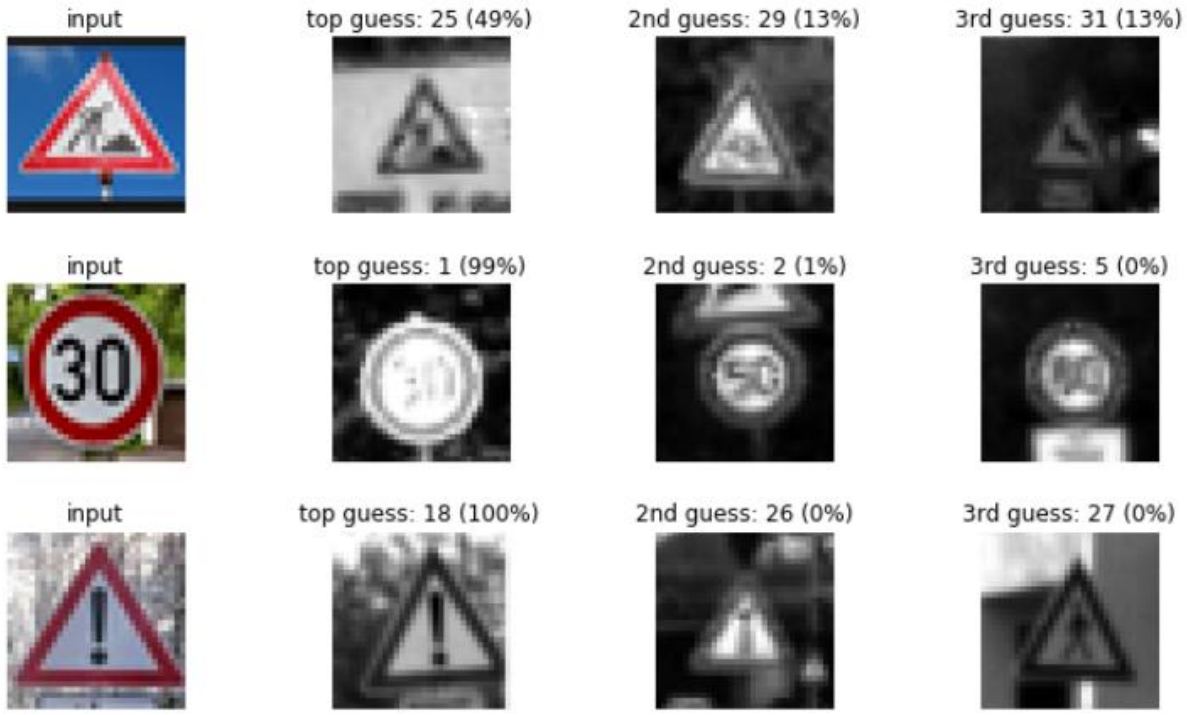
On running the CNN on the test image I got an accuracy of 83% which is quite low as only 6 images were taken into consideration.

```
INFO:tensorflow:Restoring parameters from ./lenet
Test Set Accuracy = 0.833
```

7)Model Certainity – Softmax Probabilities

The top softmax probabilities of the predictions on the captured images are outputted





This model has given the right output 5 out of 6 times for an unknown dataset images.