Traffic Sign Recognition

You're reading it! and here is a link to my [project code]

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.

I used the pandas library to calculate summary statistics of the traffic sign recognition model.

```
Number of training examples = 34799

Number of validation examples = 4410

Number of testing examples = 12630

Image data shape = (32, 32, 3)

Number of classes = 43
```

Include an exploratory visualization of the dataset.

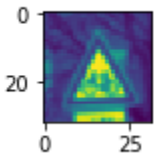
Detailed exploration is present in ipython notebook in the cell 3, 4,23 & 92

Design and Test a Model Architecture

1. Describe how you preprocessed the image data. What techniques 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. (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.)

For preprocessing I first did a gray scale conversion and followed by input normalization. For gray scale, I divided rgb image array by 3 and added all with a numpy.sum function to get an array of same shape. For input normalization, I converted number from 0 to 255 to 0 to 1. Image looks like below



I took additional data by using ImageDataGenerator from keras library. After AUGMENTATAION my training data increased double times. When I checked without AUGMENTATAION, performance of the model was not good.

```
Augumented Data= (34799, 32, 32, 3)
orignal train set= (34799, 32, 32, 3)
Combined train set= (69598, 32, 32, 3)
```

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

To the original data set I applied height shift range distortions, zoom, shear and rotational distortions to the original training data set. In the above pic, 1st from the left is the image form the original training data set and 2nd one is the augmented image.

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.

Layer	Descrition
Layer1	Convolutional. Input = 32x32x1. Output = 28x28x6
Activation	Relu

Pooling	Max pool
Layer2	Convolutional. Output = 10x10x16.
Activation	Relu
Pooling	Max pool
Layer2	Drop out – 50%
Layer3	Fully Connected. Input = 400. Output = 120.
Layer 4	Fully Connected. Input = 120. Output = 84.
Layer 5	Fully Connected. Input = 84. Output = 43.

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.

Learning rate I used is 0.001 with batch size of 34. I choosed batch size number to be a whole number with respect to the total number of image set. When I increased the batch size, I saw my validation is lessor than training accuracy. I stopped training when my training and validation accuracy crossed more than 95%.

EPOCHS = 40

BATCH_SIZE = 34

rate = 0.001

Sigma = 0.1

I quit training when my accuracy crossed more than 95%

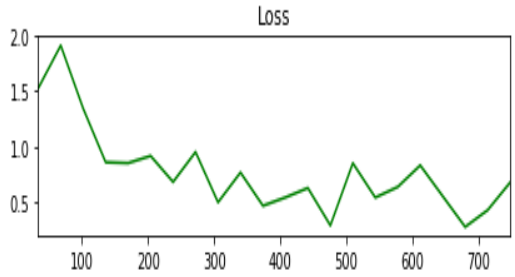
```
# over training the model
if (validation_accuracy >0.95) and (training_accuracy > 0.95):
    break
```

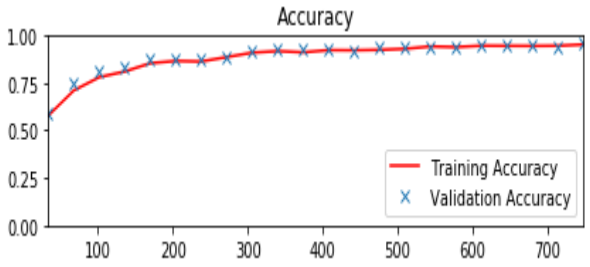
Describe the approach taken for finding a solution and getting the validation set accuracy to be at least 0.93. Include in the discussion the results on the training, validation and test sets and where in the code these were calculated. Your approach may have been an iterative process, in which case, 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.

I achieved the desired accuracy in lesser EPOCHs.

```
EPOCH 22 ...
Training Accuracy = 0.951
Validation Accuracy = 0.956
Calculate Loss = 0.676

Trained Model saved
```





Accuracy improved because of,

1. Adding drop out to the network
2. Adding Augmented data set

I included an exploratory visualization of test data in cell #23

Test accuracy I got is

Test the images with the given Test data

```
]:
import tensorflow as tf


with tf.Session() as sess:
    saver.restore(sess, save_file)

    test_accuracy = evaluate(X_test_normalized, y_test)
    print("Test Accuracy = {:.3f}".format(test_accuracy))
```

```
INFO:tensorflow:Restoring parameters from ./traffisign_model.ckpt
Test Accuracy = 0.940
```

I am using **Lenet** architecture for the current problem which is straightforward and simple to learn.

My final model results were:

Training set accuracy of 95%

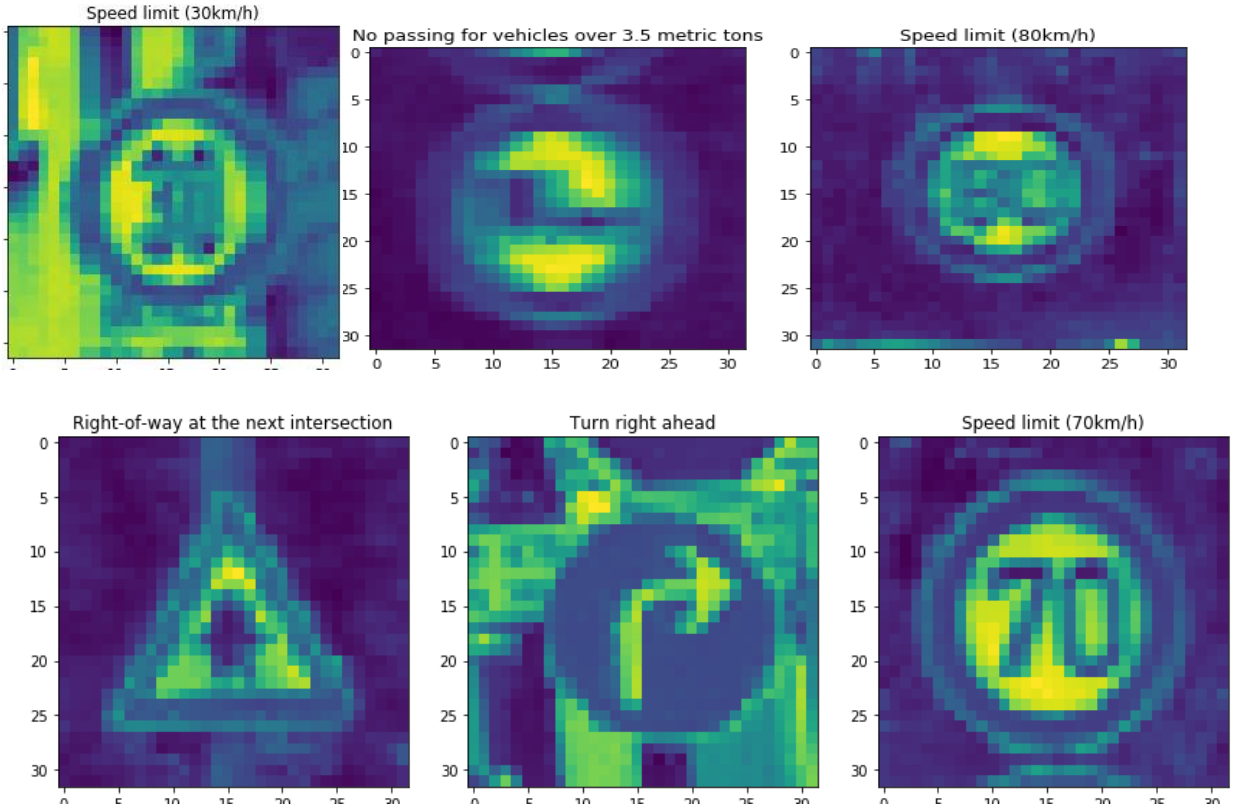
Validation set accuracy of 95.%

Test set accuracy of 94%

Test a Model on New Images

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.

Below are the 6 images from the german data set.



These images are visually little challenging to identify the traffic sign.

Image	Prediction
Speed Limit (30km/h)	100%
No passing for vehicles over 3.5 metric tons	100%
Speed limit (80km/h)	100%
Right of way at the intersection	100%
Turn Right ahead	100%

The model was able to correctly guess 5 of the 5 traffic signs, which gives an accuracy of 100%. This compares favorably to the accuracy on the test set of 94%.

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 94 and 95 cells of the Ipython notebook.

Below Output present in the ipython notebook compares prediction against each of the test sign with all the 43 possible signs. The result show 1.00 if the match is good otherwise 0 if the match is not good. Detailed exploration image is available in the ipython note book.

Example