Traffic Sign Recognition

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 individually and describe how I addressed each point in my implementation.

Writeup / README

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 code cell 5 of the notebook.

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

- The size of training set is ?
- The size of test set is ?
- The shape of a traffic sign image is ?
- The number of unique classes/labels in the data set is ?

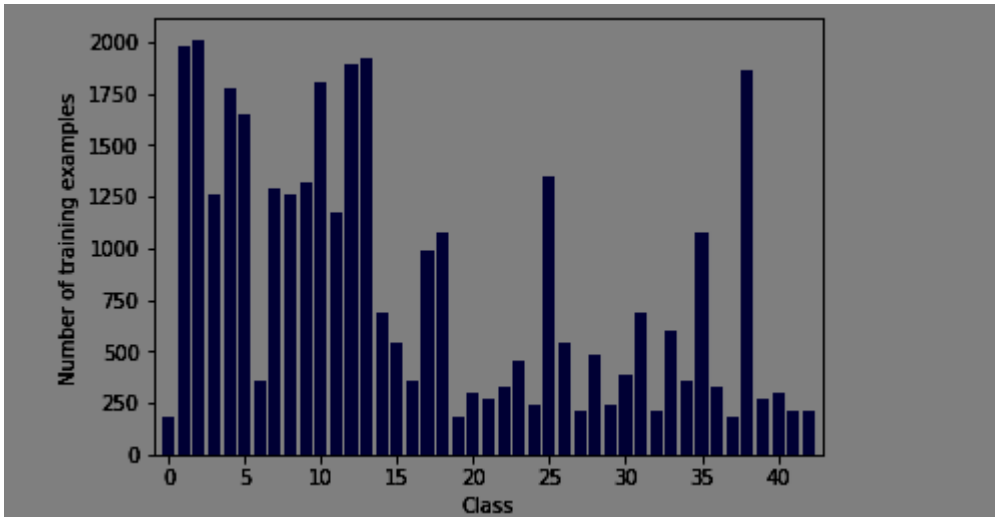
Image Shape: (32, 32, 3)
Image Shape after Grayscale: (32, 32)
Number of Class are: 43
Training Set: 34799 samples
Validation Set: 4410 samples
Test Set: 12630 samples

- 43 Clases

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 code cell 6 of the IPython notebook.

Here is an exploratory visualization of the data set. It is a bar chart showing how the data ...



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.

The code for this step is contained in the code cells(4,8) of the IPython notebook.

As a first step, I decided to convert the images to grayscale then feature scaling to reduce noise of images.

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

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)

Data set that I download from Udacity class here: https://goo.gl/OixuBM

(lesson 11 page 12) I found data is three sets train.p valid.p test.p

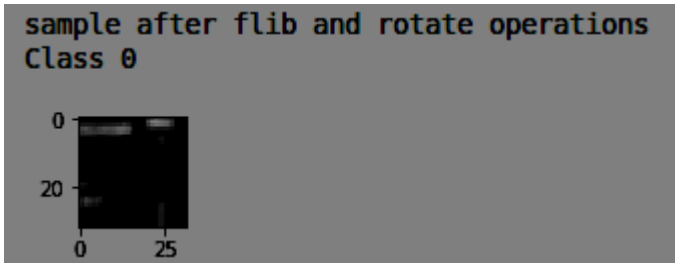
Training Set: 34799 samples
Validation Set: 4410 samples
Test Set: 12630 samples

The code cells (9,11) of the IPython notebook contains the code for augmenting the data set.

I decided Generate Fake data by flib some classes with put in consideration some classes when fliping wil be move to another classes category like classes 19,20,33,36,,37,38,39 (most classes with left or right) to increase train set.

then generate fake data for some of low counts classes by doing some rotation

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 ...

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 cell 15 of the ipython notebook.

My final model consisted of the following layers:

- Input: 32*32*1
- Conv layer 5*5 (Valid Pad, 1 stride)
- RELU
- Max pool (Valid pad)
- Conv layer 5*5 (Valid Pad, 1 stride)
- RELU
- Max pool (Valid pad)
- flatten
- FC1
- RELU
- FC2
- RELU
- FC3 logits

I played with lenet archetecure but with using Regularization (Dropout, Early stop)

Drop out:

Conv1 = keep prop 0.9

Conv2= Keep prop 0.8

Fully connected FC1 = 0.7

Fully connected FC2= 0.7

Early stop:

once no improvement in validation accuracy through > 9 epochs model will stop to avoid overfitting

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 cell 19 of the ipython notebook.

To train the model, I used an loop to pass and backpropagate for all train set (divide on batches every batch 128) for 100 epoch.

5. Describe the approach taken for finding a solution.

The code for calculating the accuracy of the model is located in cell 22 of the Ipython notebook.

I used similar technique used in lenet lab , but with add conditions to early stop model to avoid over fitting so each time with new epoch if validation accuracy improved will print small * beside accuracy result, and if validation accuracy not improved will not put it and count +1 once count > 9 model well stop.

The final test set accuracy 92.2

Note: I have some tuning and changing in architecture can increase it to 97-99.5 (I tried it with some similar projects) but this tuning and applying iterate approaches need a lot of time and not have time to finish projects 3,4,5 (Reason behind late is : Bank waste one month to produce Internet card for me because AWS told me that I cant access EC2 without this card. I worked for 10 hours daily to can finish this project)

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:

'Speed limit (20km/h): class is only one that have some rotation

‘Children cross’ little dark but not too much if it dark to much it would affect accuracy of prediction or at least reduce its probability

other pictures are clear somehow and have good quality that will not affect prediction process

I found big dataset for germany traffic sign on the germany Institute for Neural Computation, (http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset#Downloads)

dataset have thousands of images ofr each class so I just select random 5 of them with size 30*30 (dont found 32*32*3) and pad them with zero borders to become 32*32*3 and apply same preprocess operations to become 32*32*1

cell 23

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 cells 25,26 of the Ipython notebook.

Here are the results of the prediction:

```
True      ['Speed limit (20km/h)' 'Speed limit (30km/h)' 'Stop' 'Children crossing'
 'Keep right']
Predicted ['Speed limit (20km/h)' 'Speed limit (30km/h)' 'Stop' 'Children crossing'
 'Keep right']
```

The model was able to correctly guess 5 of the 5 traffic signs, which gives an accuracy of 100%. while on test set was 92,2% maybe because no dark contrast in new images (this lead to misclassified sometimes)

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 cell 27 (last code cell) of the Ipython notebook.

The top five soft max probabilities were

```
TopKV2(values=array([[ 9.99941349e-01,   4.99665257e-05,   7.36317816e-06,
          7.84437020e-07,   3.98280150e-07],
       [ 9.99677777e-01,   2.55233055e-04,   5.24519164e-05,
          1.45828153e-05,   3.55378171e-09],
```

[9.98184979e-01, 4.85186785e-04, 4.82022297e-04,
2.75808008e-04, 2.53100996e-04],
[9.98709321e-01, 1.18934806e-03, 7.69850521e-05,
1.72382934e-05, 3.07839764e-06],
[9.99989271e-01, 2.73049477e-06, 2.48695005e-06,
1.64960761e-06, 1.10545761e-06]], dtype=float32), indices=array([[0, 1, 6, 4, 21],
[1, 0, 4, 2, 5],
[14, 33, 17, 34, 15],
[28, 30, 23, 29, 24],
[38, 13, 5, 2, 1]], dtype=int32))

Optional Question: Bar plot: