Traffic Sign Recognition**

New Writeup Template

Build a Traffic Sign Recognition Project

The goals / steps of this project are the following:

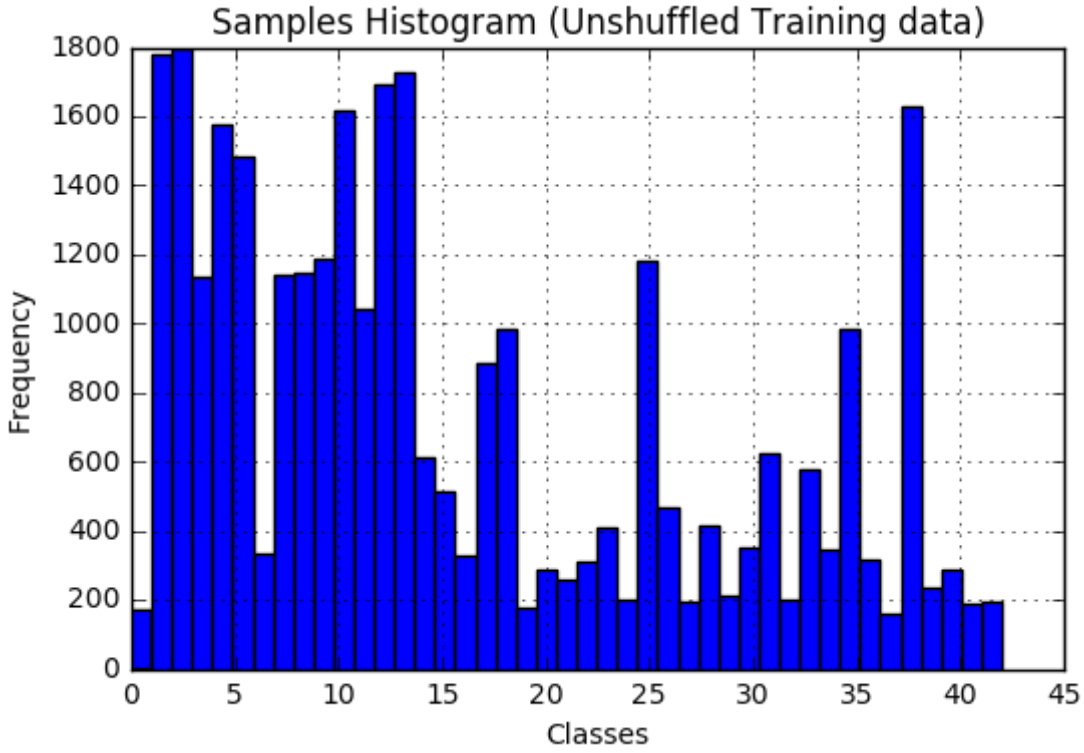
- Load the data set (see below for links to the project data set) - Done
- Explore, summarize and visualize the data set - Done
- Design, train and test a model architecture - Done
- Use the model to make predictions on new images - Done
- Analyze the softmax probabilities of the new images - Done
- Summarize the results with a written report - Done

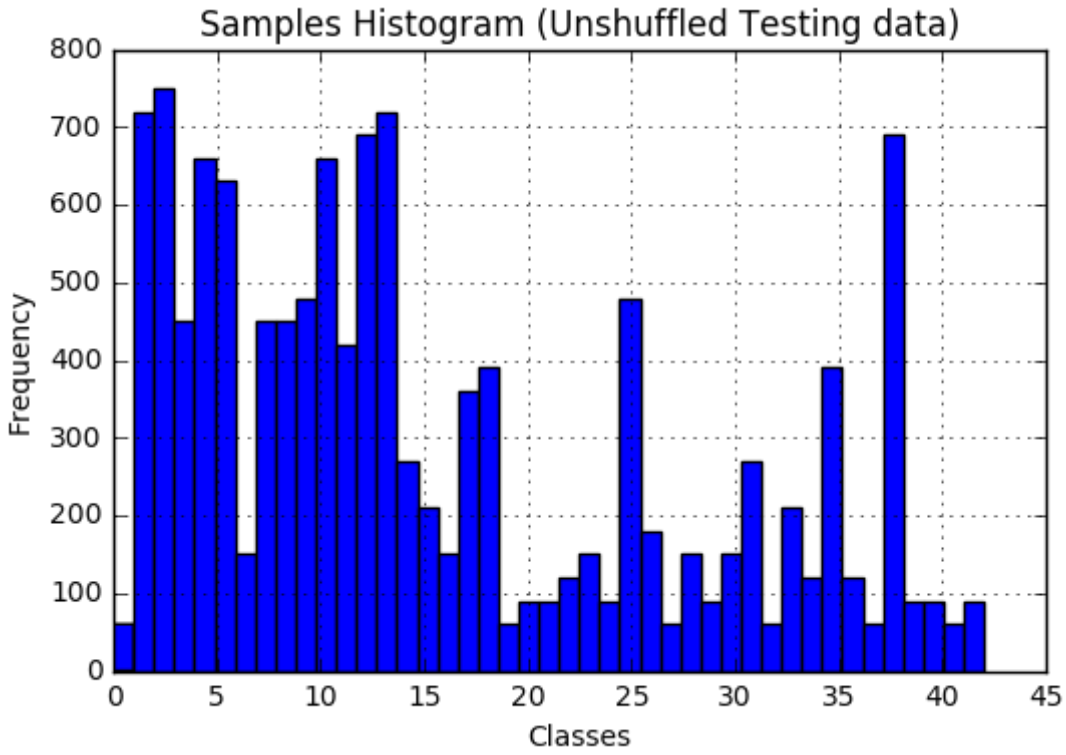
Data Set Summary & Exploration

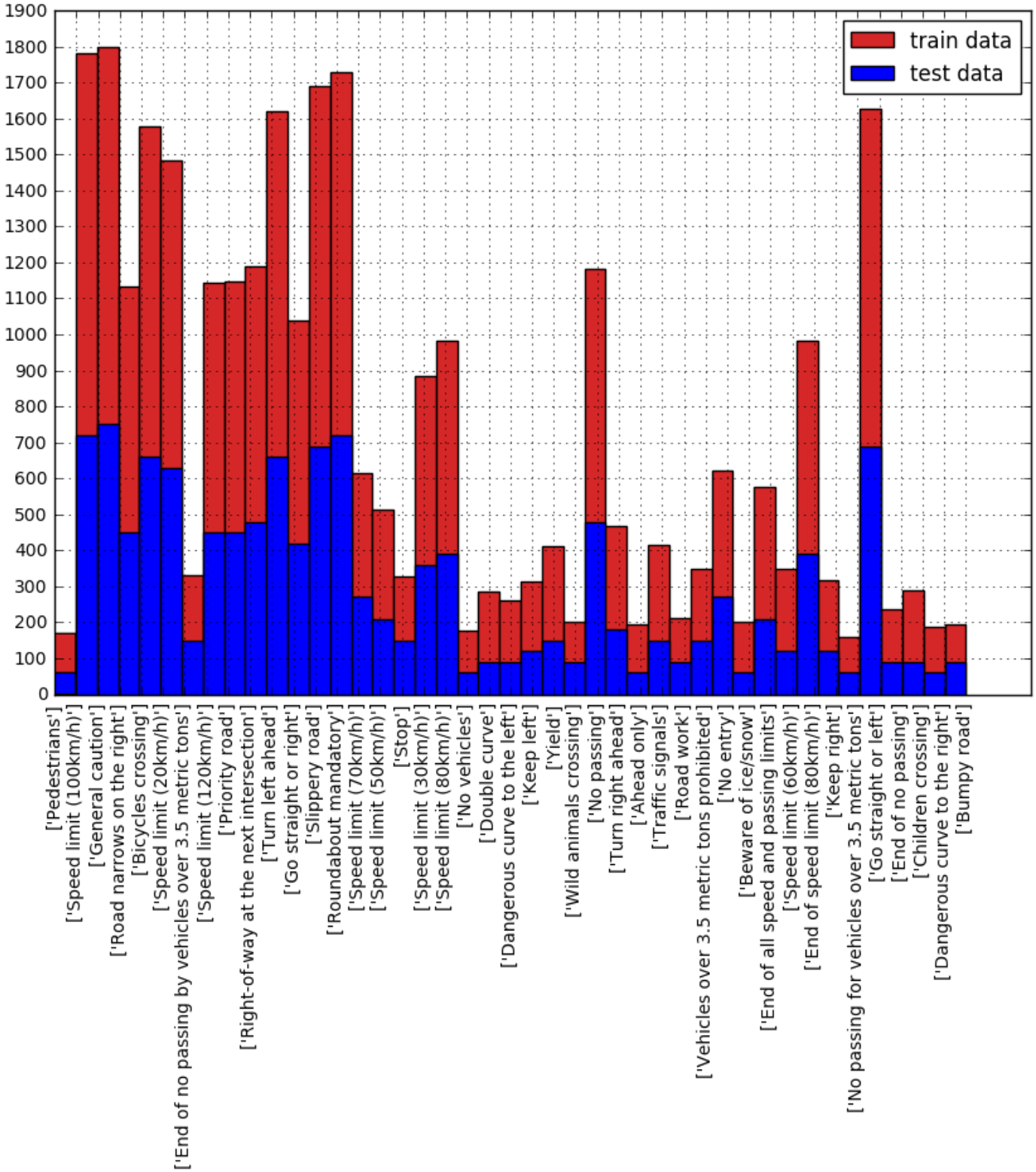
I used the pandas library to calculate summary statistics of the traffic signs data set:

- The size of training set is 31367 samples
- The size of test set is 7842 samples for the validation set and another 12630 samples for Test Set.
- The shape of a traffic sign image is 32 x 32 x 3
- The number of unique classes/labels in the data set is 43

Exploratory visualization






train data
test data
0
100
200
300
400
500
600
700
800
900
1000
1100
1200
1300
1400
1500
1600
1700
1800
1900
['Pedestrians']
['Speed limit (100km/h)']
['General caution']
['Road narrows on the right']
['Bicycles crossing']
['Speed limit (20km/h)']
['End of no passing by vehicles over 3.5 metric tons']
['Speed limit (120km/h)']
['Priority road']
['Right-of-way at the next intersection']
['Turn left ahead']
['Go straight or right']
['Slippery road']
['Roundabout mandatory']
['Speed limit (70km/h)']
['Speed limit (50km/h)']
['Stop']
['Speed limit (30km/h)']
['Speed limit (80km/h)']
['No vehicles']
['Double curve']
['Dangerous curve to the left']
['Keep left']
['Yield']
['Wild animals crossing']
['No passing']
['Turn right ahead']
['Ahead only']
['Traffic signals']
['Road work']
['Vehicles over 3.5 metric tons prohibited']
['No entry']
['Beware of ice/snow']
['End of all speed and passing limits']
['Speed limit (60km/h)']
['End of speed limit (80km/h)']
['Keep right']
['No passing for vehicles over 3.5 metric tons']
['Go straight or left']
['End of no passing']
['Children crossing']
['Dangerous curve to the right']
['Bumpy road']

Design and Test a Model Architecture

I read at the yann paper (http://yann.lecun.com/exdb/publis/pdf/sermanet-ijcnn-11.pdf) that preprocssing the images can boost your predictions upto 99 percent. So my preprocessing at first consisted solely of conversion the images to grayscale using the cv3 library but I had to use all color channels since the ConvNet could use multiple layers as its feature layers and I ended up using all the rgb values.

Throughout the modeling the training and the testing data are used as provided. This comes down to:

- Training Set: 31367 samples
- Validation Set: 7842 samples
- Test Set: 12630 samples
- Number of classes = 43

The famous LENET architecture model with 2 Convolution layers and 3 Fully connected layers was finally selected as the implementaion design. The batch size was reset to 150 due to memory limitations on my laptop. I recently asekd for gpu increase on my aws account that I created for this course and the request has not been approved yet but once the request is approved I recommend running it at Batch size of 280. This would affect the training time in general so in this report I didn't do any speedUp test and analysis.

The model has employed the following setup:

- "relu" activation,
- "dropouts" to avoid overfitting Two "pooling" layers after each convoltion layers.

The fully connected layer has 43 units for logits(network) to detect 43 different traffic signs and three channels for rgb images that are used in the training sets.

More Details on Design Decisions

Layer 1: Convolutional. Input = 32x32x3. Output = 28x28x6. strides=[1, 1, 1, 1], padding='VALID' Activation - relu Avoid overfitting- dropout Pooling - maxpool

Layer 2: Convolutional. Output = 10x10x16. strides=[1, 1, 1, 1], padding='VALID' Activation - relu Avoid overfitting- dropout Pooling - maxpool Flatten. Input = 5x5x16. Output = 400.

Layer 3: Fully Connected. Input = 400. Output = 120. Activation - relu Avoid overfitting- dropout

Layer 4: Fully Connected. Input = 120. Output = 84. Activation - relu

Layer 5: Fully Connected. Input = 84. Output = 43

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 the seventh cell of the ipython notebook and is basically a Convolutional Neural Network with two convolution layers and two fully connected layers. I used the TensorFlow tutorial here: https://www.tensorflow.org/versions/r0.11/tutorials/mnist/pros/index.html#convolution-and-pooling (https://www.tensorflow.org/versions/r0.11/tutorials/mnist/pros/index.html#convolution-and-pooling) for implementing more methods but ended up not using them for this project. I might use them later.

My final model consisted of the following layers:

Layer	Description
Input	32x32x3 RGB image
Convolution 3x3	1x1 stride, same padding, outputs 32x32x64
RELU	
Max pooling	2x2 stride, outputs 16x16x64
Convolution 3x3	Input = 400. Output = 120. Activation - relu
Fully connected	Input = 120. Output = 84. Activation - relu
Fully connected	Input = 84. Output = 43

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 eigth cell of the ipython notebook. My optimizer is the Adam Optimizer which as I learned from the course is more complex and efficient than the stochastic method that is available (https://arxiv.org/abs/1412.6980) . The SGD optimizer was used and I used a lot of the input from the class forum to better tune it. My training batch size is 128 for now and I would like to bump it up to 280 when I got my aws request approved through using EC2 instances and utilizing the GPUs there on 8 virutal nodes and 16 gb of memory but for now I am stuck with 4gb of memory and I used the 128 batch size. I set batch sizes to slightly higher values, 160 and 180, and my model was not improving accuracy. I went through over 20,000 iterations to reach over 90% accuracy on the test set. I gradually addded more and more iterations until my model got high accuracy. Other hyperparameters were a filter size of 5 x 5 pixels. The first layer has 16 filters and the second convolutional layer has 36 such filters with relu activation for both. The Fully connected size is 84.

Describe your Approach in coming up with a solution to this problem

Starting with the course material I went through a few iterations using different Conv Net configurations that I read from the papers listed in this report and also from the forum discussion. Working with the data was the data objects was not very smooth and I ran into a lot of error and overal had a high debug time. I read through anything that I could find including a tutorial by Magnus Erik Hvass Pedersen at # https://github.com/Hvass-Labs/TensorFlow-Tutorials/blob/master/02_Convolutional_Neural_Network.ipynb (https://github.com/Hvass-Labs/TensorFlow-Tutorials/blob/master/02_Convolutional_Neural_Network.ipynb). Applying the German traffic sign data in full color range to these methods was a try and error for the most part and what I could see was working and was increaseing the accuracy I picked on and used here.

Test of the 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:

![alt text][image4] ![alt text][image5] ![alt text][image6] ![alt text][image7] ![alt text][image8]

The images I used mostly have background images or other objects within the photo and that I think affected the accuracy heavily. I believe this would make classification harder especially since I didn't use any extra preprocessing steps. if I add few steps that reads in data and some how simplifies the background, Background detection methods or filters in opencv3 library, to my new images more like the images I used during the training stage I should be getting a higher performance.

2. Model's predictions

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

Here are the results of the prediction:

Image	Prediction
Double Curve	Class 20- Double Curve
Yield Sign	Class 21- Bumpy Road
Road branches in two	Class 29- Keep Left
Speed Limit 70 km/h	Class 22- Speed Limit 20 km/h
STOP Sign	Class 40- Double Curve
Do Not Enter Sign	Class 23- Road Narrows
Don't Turn Right	Class 8- Do Not Speed Sign
Speed Limit 70 km/h	Class 31-Speed Limit 70 km/h
STOP Sign	Class 14- U-turn
Speed Limit 50 km/h	Class 32- Speed Limit (50 km/h)

The model was able to correctly guess 1 of the 10 traffic signs, which gives an accuracy of 10%. But it could predict almost 100 percent of all Speed Limit signs but failed to predict the right top speed for these signs. I think the proportion of the sign to the total size of the image is an important fact and images should be preprocessed and resized before being tested by the model to achive higher accuracy. This makes sense since most of the images that were used during the training had similar value for their area of the sign ove the area of the image (more than 50 percent of the image is actually the area of the sign).

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 11th cell of the Ipython notebook and the output is listed here:

```
['Dangerous curve to the right'] 0.0396727
['Double curve'] 0.0428837
['Bicycles crossing'] 0.0350074
['Bumpy road'] 0.0340346
['Roundabout mandatory'] 0.0385667
['Slippery road'] 0.0397616
['Speed limit (120km/h)'] 0.0433578


['Traffic signals'] 0.0380962
['Bumpy road'] 0.0404578
['Double curve'] 0.0347044
['Keep left'] 0.0338381
['Speed limit (120km/h)'] 0.0341477
['Double curve'] 0.0369758
['End of no passing'] 0.0411868


['Road narrows on the right'] 0.0372592
['Go straight or left'] 0.0366764
['Ahead only'] 0.0334628
['Roundabout mandatory'] 0.0334439
['Slippery road'] 0.0338212
['Speed limit (20km/h)'] 0.0328545
['Keep left'] 0.0387621


['Speed limit (20km/h)'] 0.0367148
['Dangerous curve to the right'] 0.0315494
['Roundabout mandatory'] 0.0316039
['Go straight or right'] 0.0323814
['Speed limit (30km/h)'] 0.0338054
['Roundabout mandatory'] 0.032706
['Traffic signals'] 0.0347361


['No passing'] 0.0318367
['Speed limit (30km/h)'] 0.0299097
['Speed limit (120km/h)'] 0.0309801
['Speed limit (70km/h)'] 0.0281022
['Road narrows on the right'] 0.0320837
['Speed limit (30km/h)'] 0.0305911
['Double curve'] 0.0342101
```

For the first image, the model is relatively sure that this is a stop sign (probability of approximately of 0.6), and the image does contain a stop sign. The top five soft max probabilities were

Probability	Prediction

.41	Double Curve
.40	End of no passing (~ Bumpy Road)
.39	Keep Left
.38	Speed Limit (20 km/h)
.34	Double Curve

References and Appendix:

1. CS231 Cornell University Library, Convolutional Neural Network for Image Recognition http://cs231n.github.io/ (http://cs231n.github.io/)
2. Caffe Link (http://caffe.berkeleyvision.org/) Especially lessons learned from current tested models for exmpale the work from Karpathy (http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/)

Section: 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 of the ten German traffic signs that I found on the web:


70


50