#**Traffic Sign Recognition**

Build a Traffic Sign Recognition Project

Rubric Points
Here I will consider the [rubric points](https://review.udacity.com/#!/rubrics/481/view) individually and describe how I addressed each point in my implementation.

** (German and Belgium) Traffic sign Recognition

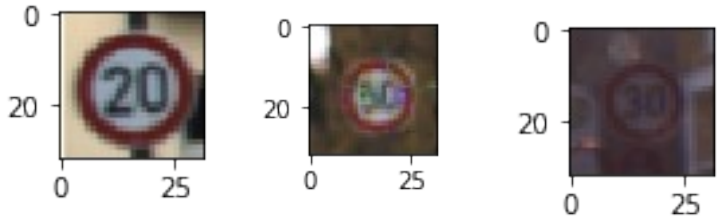
Data Set Summary & Exploration

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.

The summary statistics of the traffic
signs data set:

* The size of training set is	40471 example
* The size of the validation set is	5119 example
* The size of test set is	13339 example
* The shape of a traffic sign image is	(32,32,3)
* The number of unique classes/labels in the data set is	105

####2. Include an exploratory visualization of the dataset.




```
class 0 = 180 image......percent of the training set= 0.44476291665637124 %
class 1 = 1980 image......percent of the training set= 4.892392083220083 %
class 2 = 2010 image......percent of the training set= 4.966519235996145 %
```

###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.)
Data Pre-processing:
Firstly, converted the images to gray scale,
Then I normalized the image data to get all the data on the same scale,
And finally I shuffled the data to cancel any effect that may come from the order of the training set

####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:

Layer	Discription
Input	32x32x3 RGB image
Convolution 5x5	1x1 stride, Valid padding, outputs 28x28x32
RELU	...
Average pooling	2x2 stride, outputs 14x14x32
Convolution 5x5	1x1 stride, Valid padding, outputs 10x10x64
RELU	...
Average pooling	2x2 stride, outputs 5x5x64
Fully connected	Input: 5*5*64 outputs:300 neurons
Sigmoid	...
Fully connected	Input: 300 outputs:150 neurons
Sigmoid	...
Classifier	Inputs: 150 outputs: 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.

To train the model, I used hyperparameters:
got them by trying and watching the accuracy changes

1. 2 convolution filters of 5x5 strides=1x1
2. Average pooling strides 2x2 (choose average to get more accuracy)
3. 2 fully connected layers with outputs 300,150
4. 2 Sigmoid activation functions with the fully connected layers
5. number of epochs = 10
6. batch size= 256
7. learning rate=0.001

####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 95%
* test set accuracy of 94%

If a well known architecture was chosen:
* What architecture was chosen?
LeNet, used the Sigmoid activation functions in the last 2 layers to get its great error correction advantage without falling into the its vanishing gradient disadvantage

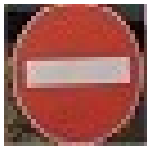
* Why did you believe it would be relevant to the traffic sign application?
Because it was recommended by many professionals on the Internet

* How does the final model's accuracy on the training, validation and test set provide evidence that the model is working well?
The training gives a very low validation accuracy at the beginning but it converges fast that it hits the 90% by the 4th epoch and stabilizes at 95% by the 10th epoch

###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:

####2. 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).

Here are the results of the prediction:

Image	Prediction
No entry	BelgiumTS- class 22
Turn right ahead	Turn right ahead
No passing for vehicles over 3.5 metric tons	No passing for vehicles over 3.5 metric tons
Children crossing	BelgiumTS - class 7
Stop sign	BelgiumTS- class 41

The model was able to correctly guess 2 of the 5 traffic signs, which gives an accuracy of 40%. This compares favorably to the accuracy on the test set of 94%

####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. 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 13th cell of the Ipython notebook.

For the first image, the model is relatively sure that this is a Belgium class 22 ahead sign (probability of 0.96), but it is No entry. The top five soft max probabilities were

Probability	Prediction
0.96	BelgiumTS class 22
0.016	BelgiumTS class 35
0.003	BelgiumTS class 39
0.0024	No entry
0.0021	Stop

For the second image, the model is relatively sure that this is a Turn right ahead sign (probability of 0.81), and the image does contain a Turn right ahead sign. The top five soft max probabilities were

Probability	Prediction
0.81	Turn right ahead
0.035	Stop
0.030	Priority road
0.029	Right-of-way at the next intersection
0.013	BelgiumTS class 39

For the third image, the model is relatively sure that this is a No passing for vehicles over 3.5 metric tons sign (probability of 0.65), and the image does contain a no passing by vehicles over 3.5 metric tons sign. The top five soft max probabilities were

Probability	Prediction
0.65	no passing by vehicles over 3.5 metric tons
0.086	End of no passing by vehicles over 3.5 metric tons
0.029	BelgiumTS class 37
0.027	BelgiumTS class 31
0.023	Slippery road

For the forth image, the model is relatively sure that this is BelgiumTS- class 7(probability of 0.38), but it is a Children crossing sign. The top five soft max probabilities were

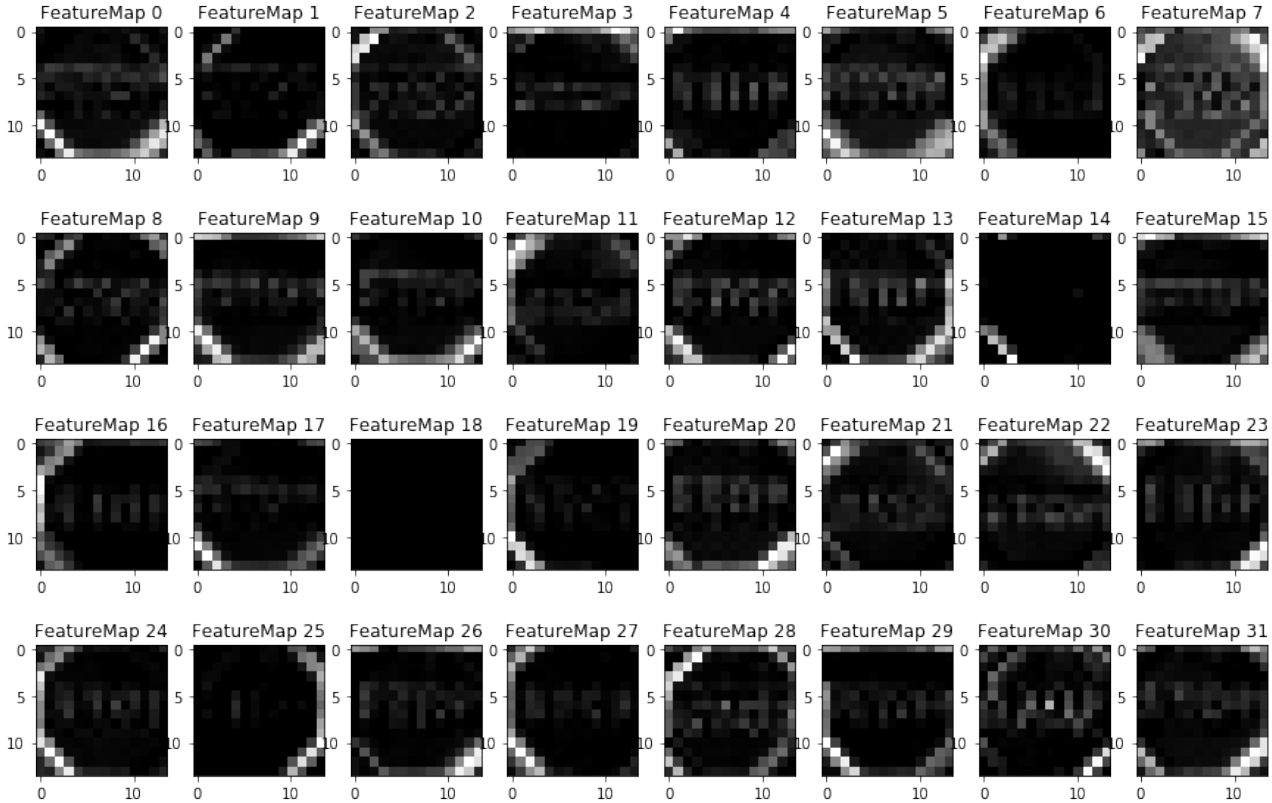
Probability	Prediction
0.38	BelgiumTS class 7
0.098	BelgiumTS class 81
0.066	BelgiumTS class 9
0.064	BelgiumTS class 57
0.063	BelgiumTS class 8

For the fifth image, the model is relatively sure that this is a stop sign (probability of 0.83), and the image does contain a Stop sign. The top five soft max probabilities were

Probability	Prediction
0.409	BelgiumTS class 41
0.29	BelgiumTS class 42
0.068	BelgiumTS class 21
0.052	Stop
0.040	Priority road

(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?



The above filter images are the out of the first convolution layer for a Stop sign … it used the edges in the image thus detecting the white circle and the edges of the word Stop.

Project version 1:

The pre-processing:

just Normalizing

Results:

Validation accuracy: 97%

Testing accuracy: 94%

Prediction results:

5 out of 5 images predicted correctly

Project version 2:

The pre-processing:

Converting to gray scale

Normalizing

Results:

Validation accuracy: 97%

Testing accuracy: 94%

Prediction results:

5 out of 5 images predicted correctly

We got the same results using gray scaled images, that means that the color features doesn't add to the power of the predictions, so by converting to gray scale we're dealing with only 1 channel instead of 3 channels , thus reducing the needed computational power to train the model

Adding the Belgium dataset to the German dataset and feed them to "Project version 2"

Belgium dataset source: http://www.vision.ee.ethz.ch/~timofter/traffic_signs/

the files labeled: BelgiumTS for Classification (training and testing)

- Combined them together into the training folder to make my own splits (80% 10% 10%)
- Inside the training folder, a sub-folders 00000 to 00061 each represents a class of images (.ppm)
- Added another sub-folders called "old" consists of the original images and "resized" folder consists of the 32x32 resized images (.jpg)
- Ran the Python code "Adding new data.py" which resizes the original data, makes X and Y arrays of them, splits the arrays , concatenates the X,Y to the original German dataset , finally save the new combined dataset to new pickle files
- Import the new dataset to the Ipython Notebook code

Results:

Validation accuracy: 95%

Testing accuracy: 94%

Prediction results:

2 out of 5 images predicted correctly

A major drop, but WHY?

Because of 2 points:

1- The Belgium dataset consists of 7090 images (62 classes) only while the german consists of 51830 images (43 classes). That makes the Belgium data hard to be detected resulting in a drop in the accuracy.

2- There are some similarities between the datasets, so you find the same sign with 2 different labels.

So combining the 2 data sets need more data which I don't have and more work on matching the 2 sets and some traffic signs knowledge