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) -Completed

Explore, summarize and visualize the data set -Completed

Design, train and test a model architecture -Completed

Use the model to make predictions on new images -Completed

Analyze the softmax probabilities of the new images -Completed

Summarize the results with a written report -Completed

Rubric Points

Write-up / README

###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.

I used a couple of the built-in libraries to python and the numpy library extensively to calculate summary statistics of the traffic signs data set. The German dataset did not have a validation set, so I used the sklearn library to split the dataset.

Pre-Split

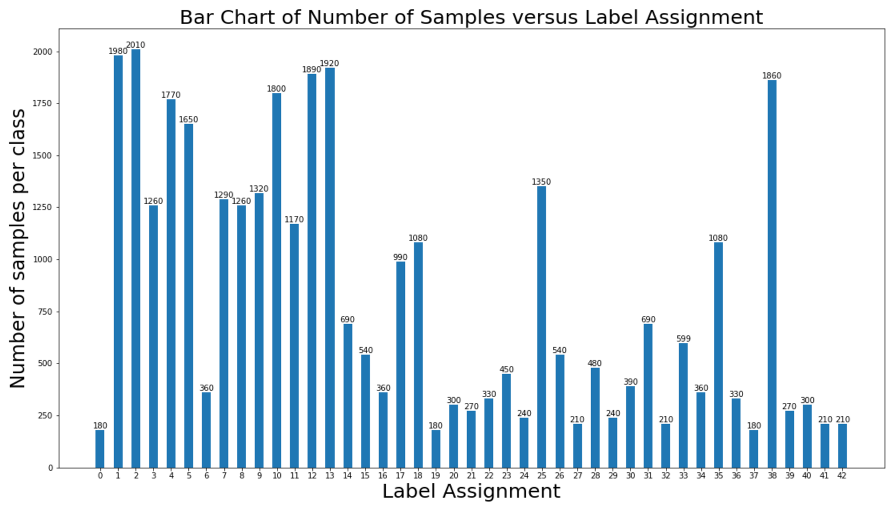
The size of training set is 34799

The size of the validation set is 0

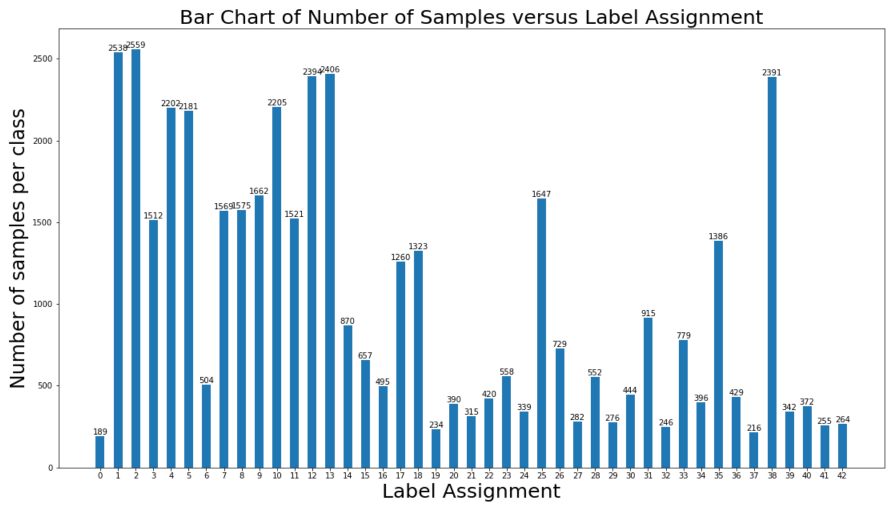
The size of test set is 12630

The shape of a traffic sign image is 32,32,3

The number of unique classes/labels in the data set is 43 *(calculated by using the set command to identify unique values and then find the len of the result.)*



I also realized that there were certain classes that had a very low number of images available.
So After going through a few Data Augmentation methods, I decided to go with Increases the images by taking 1000 random images and rotating them about a set of 10 angles between -45 degrees and +45degrees. The following images Is the number of images that were increased. For bigger pictures please look at the code supplied in the ipynb file. All Classes has an increase in the images present.

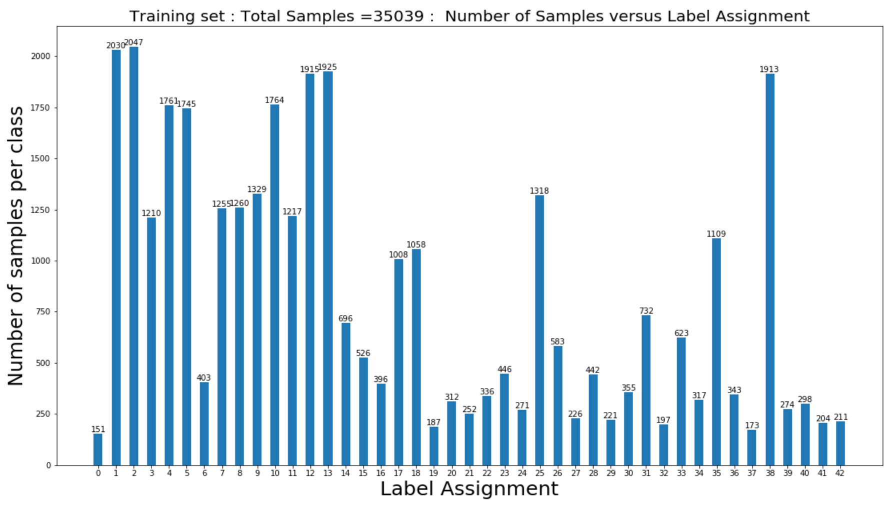


####2. Include an exploratory visualization of the dataset.

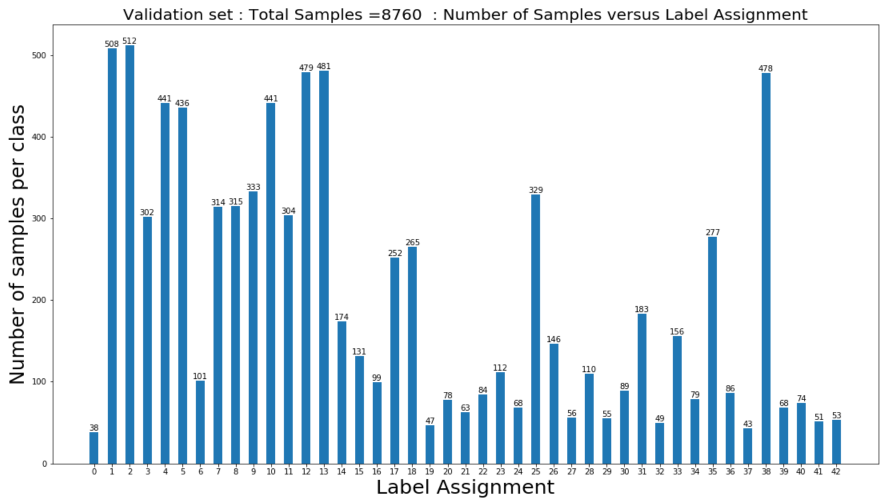
Post-Spli

Post split, Here is the visualization of the dataset.

<u>Training</u>



<u>Validation</u>



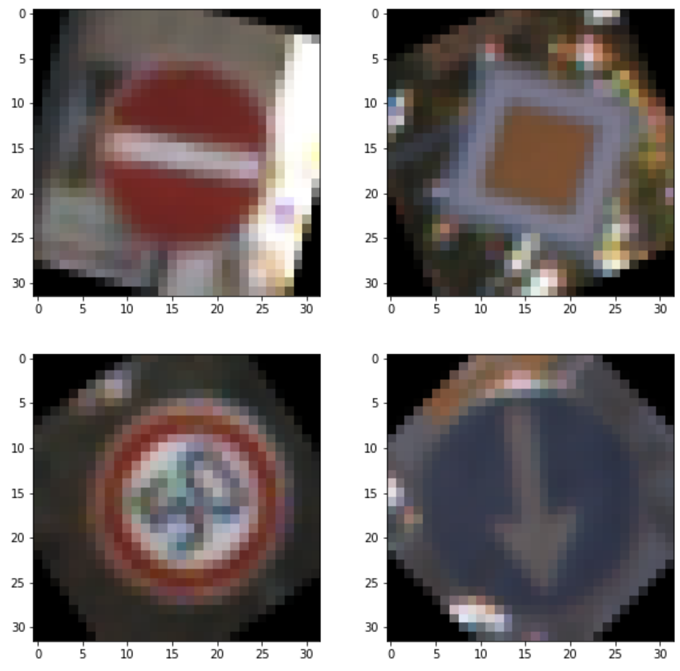
There are multiple excel files present in the github repo that are results from multiple simulations.

###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.)

As a first step, I also realized that there were certain classes that had a very low number of images available.
So After going through a few Data Augmentation methods, I decided to go with Increases the images by taking 1000 random images and rotating them about a set of 10 angles between -45 degrees and +45degrees. The following images Is the number of images that were increased. For bigger pictures please look at the code supplied in the ipynb file. All Classes has an increase in the images present.

Sample of Augmented (rotated) Images

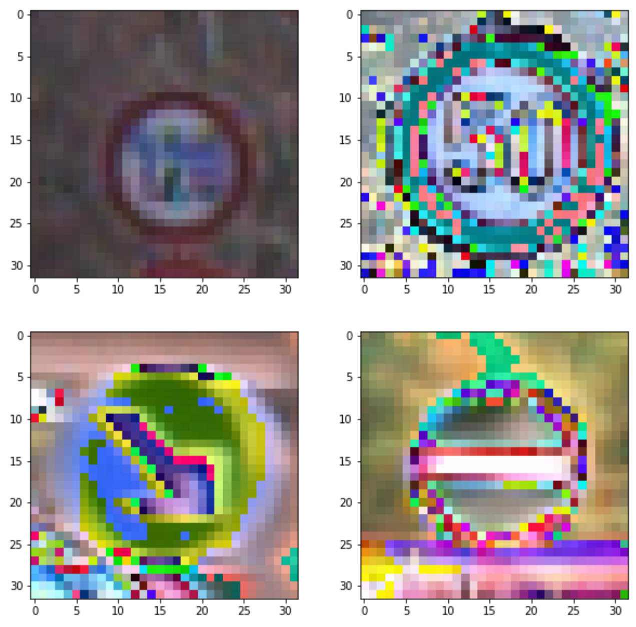


During my Pre-processsing tests, I found something very odd. If I tried splitting the data first and then running the training, validation and test set separated through the same function, they all did not return the same results. The Training accuracy and the validation accuracy all dropped to 5 percent accuracy.

I attempted to run multiple forms of preprocessing.

1) Normalization *(RGB-128/128 – method provided in class)*
2) Zero centering the data and then Normalizing *(http://www.kdnuggets.com/2016/03/must-know-tips-deep-learning-part-1.html)*
3) PCA Whitening *(http://www.kdnuggets.com/2016/03/must-know-tips-deep-learning-part-1.html)*
4) Standard normalization (using mean and standard deviation over pixels in image)
5) Transform to Grayscale, HSV, Canny, and concatenation of the three.
   Unfortunately no one on the forums was able to help me understand why and my mentor was unsure either. Finally I found that I was able to do the first type of normalization before splitting the data so I decided to move on cause I was already stuck for quite a few days and was wasting time. I could always get back to finding the reason after submitting the project.

Sample of Normalized images



####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 a very slightly modified LeNet Structure.

Layer	Description
Input	32x32x3 RGB image
Filter	5x5x25
Convolution 3x3	1x1 stride, same padding, outputs 28x28x25
RELU	
Max pooling	2x2 stride, outputs 14x14x25
Convolution 3x3	1x1 stride, same padding, outputs 10x10x50
RELU	
Max pooling	2x2 stride, outputs 5x5x50
Flatten	1250
Fully connected	120
RELU	
Dropout	
Fully connected	84
RELU	
Dropout	
Fully connected	43

Finally Softmax probabilities on the final test model.

####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.

I initially tried adding additional dropout layers after the first two convolution layers and their relus but that seemed to bring the accuracy down by quite a bit. So I removed them.
Then I tried chaning the sigma to multiple values between 0.1 to 1.2. The higher I went, the lower my accuracy seemed to be which corresponds with the class in saying that you want to keep the weights and the bias to a relatively small mean and std.

After that I tried altering batch size. MY internal GPU on my laptop was 2GB. I tried a batch size of 128 initially and my computer tended to bluescreen.

So calculating the number of bytes my computer would need to process all the information, I brought down the batch size to 50.

After than I attempted to alter the learning rate. But decreasing the Learning by a factor of negative 10 from the original 0.001 gave me steady but very slow progress which, granted, that the number of tests I would need to run before then and the final submission would be many, I decided against it. Also, seeing as my accuracy was already 94 percent plus and rising, I decided to leave it at that.

####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 Accuracy	= 0.971
Validation loss	= 0.212
Training Accuracy	= 0.999
Training loss	= 0.042

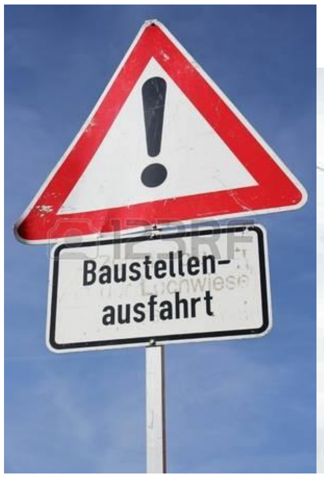
Staggered simulation results are shown in the github repo excel sheets.

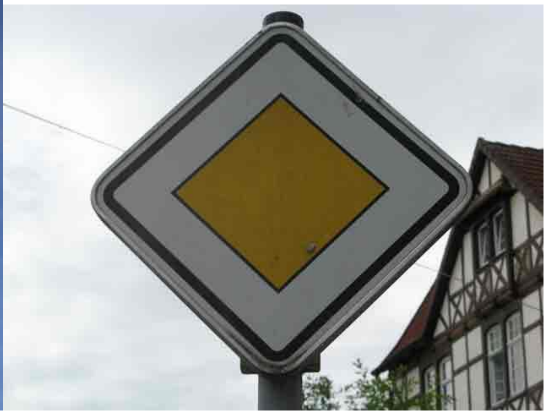
After running LeNet architecture, I wanted to test out implement a combination of Inception module followed by the LeNet architecture to see if it would work. But I kept running out of time since I was stuck on the Pre-processing bit for most of the project.

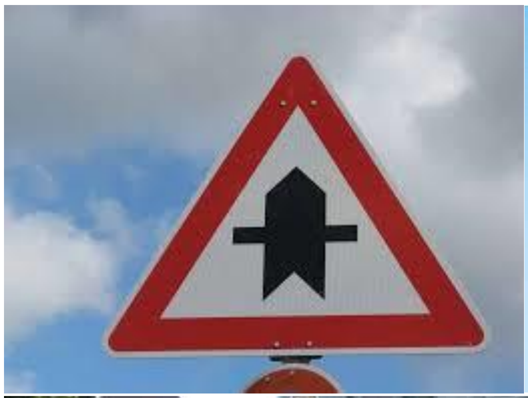
###Test a Model on New Images

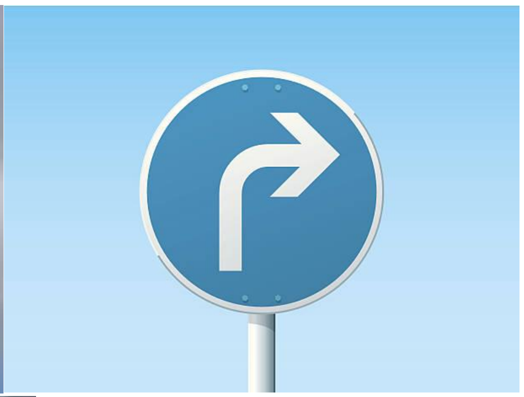
####1. Choose five German traffic signs found on the web and provide them in the report. For each image, discuss what quality or qualities might be difficult to classify.

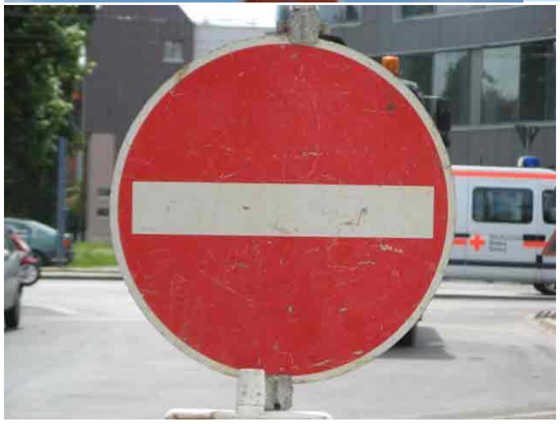
Here are five German traffic signs that I found on the web:











The first image might be difficult to classify because of the hologram across the image as well. Multiple images on Google seemd to be setup with gettyimages hologram across it which I thought would interfere with the Normalization process.

####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:

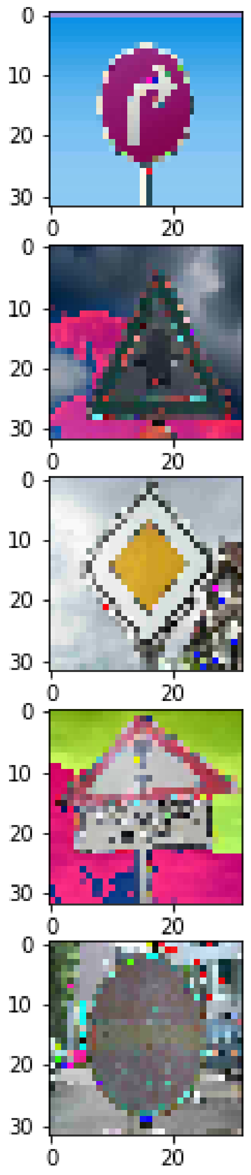
Image Prediction

```
Correct predictions
Predicted Class:   Priority road
Class Confidence: 1.0

Predicted Class:   Right-of-way at the next intersection
Class Confidence: 1.0

Predicted Class:   No entry
Class Confidence: 1.0

Incorrect predictions
Predicted Class:   Speed limit (120km/h)
Class Confidence: 0.418383
Actual Class: General Caution

Predicted Class:   General caution
Class Confidence: 0.673979
Actual Class: Right turn ahead.
```

The model was able to correctly guess 3 of the 5 traffic signs, which gives an accuracy of 60%. This compares inaccurately to the accuracy of the validation set which provided an accuracy of 97 percent.

I’m not sure why it didn’t predict the images correctly.

Final tests images after normalization.



####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 top five soft max probabilities were

Probability   Prediction

First Image - Right Turn Ahead

```
Predicted Class:   General caution          Class Confidence: 0.673979
Predicted Class:   Traffic signals          Class Confidence: 0.201113
Predicted Class:   Right-of-way at the next intersection
                                            Class Confidence: 0.103295
Predicted Class:   Road work                Class Confidence: 0.0127792
Predicted Class:   Pedestrians              Class Confidence: 0.0071179
```

Second Image – Right of way at next intersection

```
Predicted Class:   Right-of-way at the next intersection
                                            Class Confidence: 1.0
Predicted Class:   Beware of ice/snow       Class Confidence: 3.16953e-25
Predicted Class:   Traffic signals          Class Confidence: 9.06876e-26
Predicted Class:   Road work                Class Confidence: 4.84408e-26
Predicted Class:   Pedestrians              Class Confidence: 7.38746e-27
```

Third Image – Priority Road

```
Predicted Class:   Priority road            Class Confidence: 1.0
Predicted Class:   Speed limit (20km/h)     Class Confidence: 0.0
Predicted Class:   Speed limit (30km/h)     Class Confidence: 0.0
Predicted Class:   Speed limit (50km/h)     Class Confidence: 0.0
Predicted Class:   Speed limit (60km/h)     Class Confidence: 0.0
```

Fourth Image – General caution

```
Predicted Class:   Speed limit (120km/h)    Class Confidence: 0.418383
Predicted Class:   Speed limit (70km/h)     Class Confidence: 0.349812
Predicted Class:   Speed limit (100km/h)    Class Confidence: 0.231421
Predicted Class:   Wild animals crossing    Class Confidence: 0.000302915
Predicted Class:   Traffic signals          Class Confidence: 3.76852e-05
```

Fifth Image – No Entry

```
Predicted Class:   No entry                 Class Confidence: 1.0
Predicted Class:   Stop                     Class Confidence: 3.06562e-26
Predicted Class:   Bumpy road               Class Confidence: 3.43033e-28
Predicted Class:   No passing               Class Confidence: 5.0107e-29
Predicted Class:   Priority road            Class Confidence: 9.50505e-30
```