**Traffic Sign Recognition**

**Build a Traffic Sign Recognition Project**

The goal of this project is to build a neural network based learning model that is able to identify the traffic signs.

The project includes the following main steps:

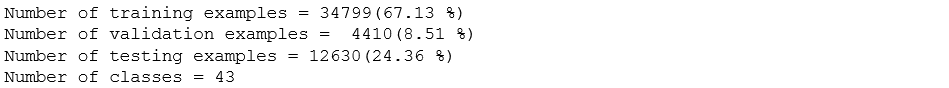
* Load the data set for [German Traffic Signs](https://d17h27t6h515a5.cloudfront.net/topher/2017/February/5898cd6f_traffic-signs-data/traffic-signs-data.zip)
* Visualize, analyze and understand the dataset
* Design, train and test a model based on LeNet-5 architecture
* Modify the model (LeNet-HI)
* Design, train and test the LeNet-HI model
* Use the LeNet-HI model to make predictions on new images
* Analyze the softmax probabilities of the new images
* Summarize the results with a written report

**Data Set Summary & Exploration**

**1. Dataset Summary**

Using python and numpy methods the following statistics of the traffic signs data set are extracted:



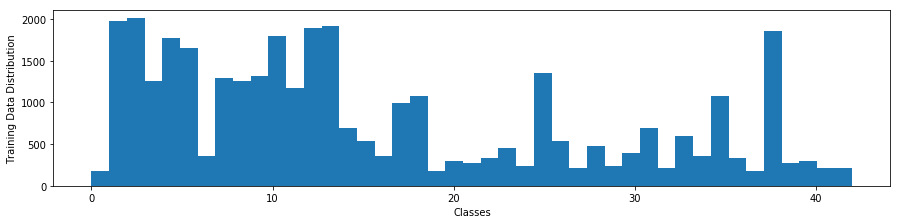


**2. Exploratory visualization of the dataset:**

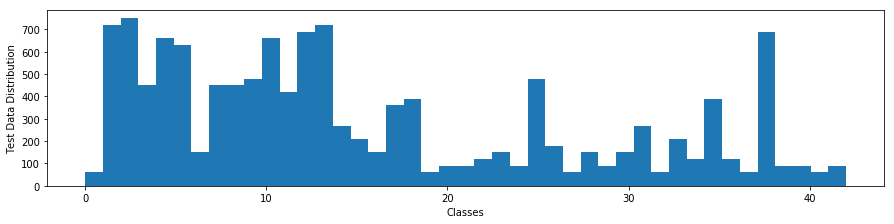
Here is an exploratory visualization of the data set. First randomly some of the pictures are shown:



Then a histogram chart shows how the data are distributed over the classes stretch in the training, validation and testing data sets:







**Design and Test a Model Architecture**

**1. Data Preparation:**

As a first step, the images are converted to grayscale to reduce the number of features for the model and make it easier for the network to learn from patterns only (not colors). Here is an example of a traffic sign image before and after grayscaling:



As a last step, I normalized the image data using the formula (pixel-128)/128, to map the data to the range of [-1, 1] instead of [0, 255] with mean value of around zero. This helps training the model because the weights will not be dominated by features with higher range. Here are the summary of the mean values for the training dataset before and after the normalization:

mean(X) mean(X\_normal)

82.677589037 -0.354081335648

83.5564273756 -0.347215411128

82.1484603612 -0.358215153428

**2. Model Architecture:**

My final model consisted of the following layers:

|  |  |
| --- | --- |
| **Layer** | **Description** |
| Input | 32x32x1 Grayscale image |
| Convolution | 5x5x1, 1 stride, Valid padding, outputs 28x28x6 |
| RELU |  |
| Max pooling | outputs 14x14x6 |
| Convolution | 5x5x1, 1 stride, Valid padding, outputs 10x10x16 |
| RELU |  |
| Max pooling | outputs 5x5x16 |
| Flatten | Output 400 |
| Dropout | keep\_probe = 0.5 |
| Fully connected | Output =120 |
| RELU |  |
| Fully connected | Output =84 |
| RELU |  |
| Fully connected | Output =43 |
| Softmax |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

**3. Model Training:**

To train the model, I used the AdamOptimizer with learning rate of 0.0009, batch\_size of 100, over 70 epochs.

Aside from playing around with the learning rate and batch size to increase the validation set accuracy, a Dropout layer was added to the LeNet-5 architecture.

Using LeNet-5 a training accuracy of 93% and testing accuracy of 94.5% is achieved. Hyperparameters are used as below: Epochs = 150, Learning Rate = 0.009, Batch Size = 100 while this performance is good, to make the network more robust and efficient the Dropout technique is used in my model (LeNet\_HI) between the convolution layers and fully connected layers (hypothesize is that this will make conv and fully connected layer learn on their own and ovoid overfitting). A keep probability of 0.5 is used. As we'll see below, with this small improvement the training accuracy to %96.4 and testing accuracy of %94.5 is achieved (at some runs the numbers were even higher).

My final model results were:

* training set accuracy of 96.4%
* validation set accuracy of 94.5%
* test set accuracy of 100% (see below)

**Test a Model on New Images**

Here are 7 German traffic signs that I found on the web:



Images 4 and 6 are a little difficult to detect because the signs are a little rotated in the image. Image 7 was not square initially (aspect ratio was not 1). Images 1 and 2 have a busy background.

The prediction accuracy is 100%. The probabilities for 2nd to 4th guesses for each image are shown below:

