# Traffic Sign Recognition

By: Jesse Turner (Self-Driving Car Engineer Program - Udacity)

---

**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

## Dataset Exploration

**Dataset Summary:**

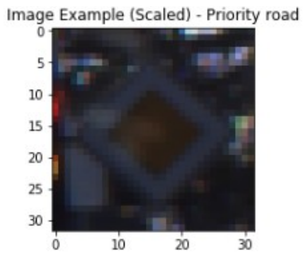
Training, testing, and validation data / labels were imported into 'float' data type numpy arrays from the provided pickle data files. Inspection of the provided data sets showed the following data set parameters:

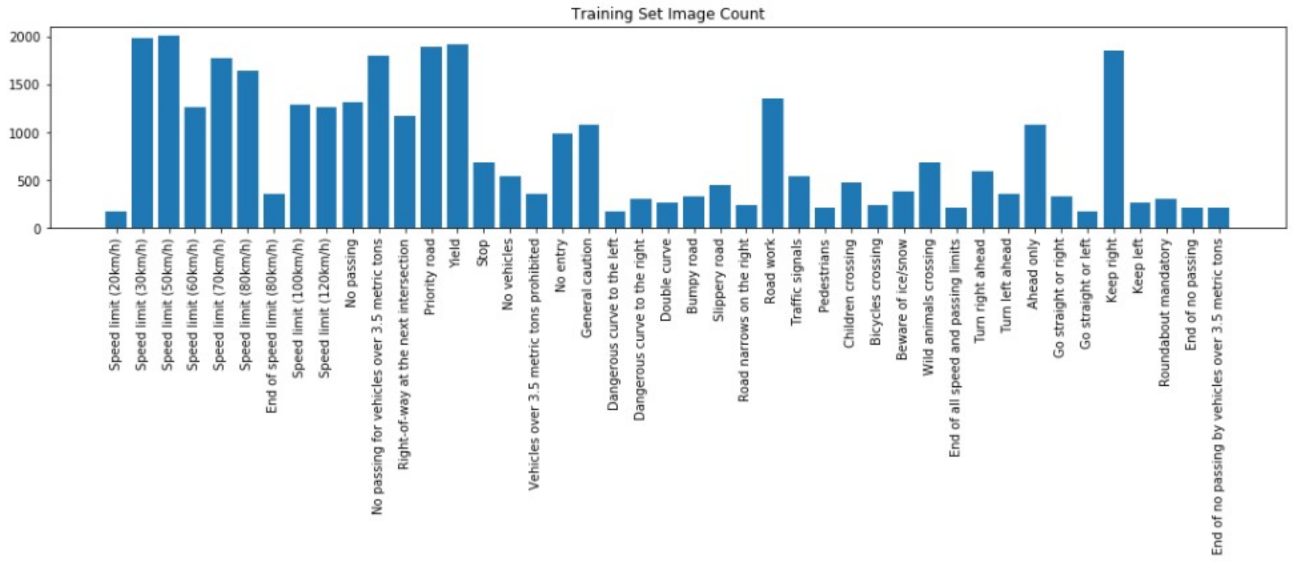
- •The size of training set is 34799 examples
- •The size of the validation set is 4410 examples
- •The size of test set is 12630 examples
- •The shape of a traffic sign image is [32,32,3] in RGB format
- •The number of unique classes/labels in the data set is 43

**Exploratory Visualization:**

Exploratory visualization of the training data set was conducted to determine the numbers of examples in the training set for each label, as well as to provide a visual representation of an random training image.

An example of the visualization is provided as follows:





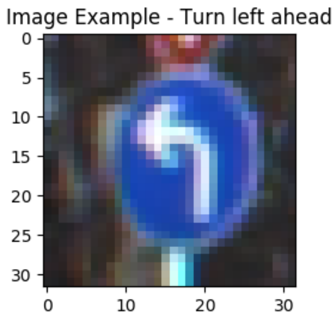
As can be seen from the above histogram, the highest number of training examples are provided for “Yield”, “Priority Road”, “Keep Right”, “No Passing for Vehicles Over 3.5 Metric Tons”, “Speed Limit 20 km/h”, “Speed Limit 30 km/h”, “Speed Limit 70 km/h”, and “Speed Limit 80 km/h”.  The histogram also shows that there are numerous training data labels which do not have strong representation in the data set, with few than 1000 or even 500 samples, as can can be seen especially on the right hand half of the histogram.

## Design and Test of Model Architecture

### Pre-Processing:

During pre-processing of the data, I scaled and normalized each image such that the each color channel will have RGB values mapped from (0,255) to (-1 to 1). This was done to reduce bias of image data as well as reduce the scale of training weights. As weights are initialized using a normal distribution with standard deviation of 0.1 and 0 bias, scaling and normalization of the data should help converge CNN weights and biases at a quicker rate.

Here is an example of a traffic sign image before and after scaling / normalization using matlibplot.imshow:



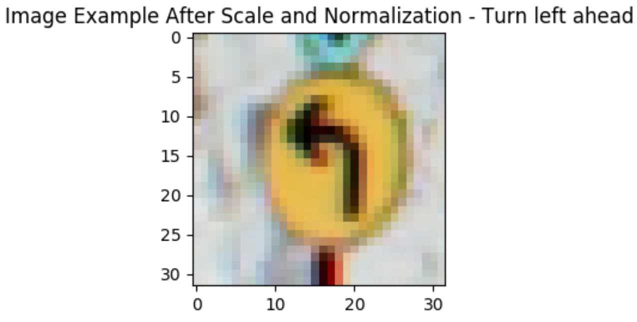


Image Example After Scale and Normalization - Turn left ahead
0
5
10
15
20
25
30
0
10
20
30

I chose not to grayscale the image, as each color channel provides additional information to the CNN in classifying each image.

**Model Architecture:**

The architecture of my model was selected based on extension of the LeNet architecture in order to maximize the depth of the CNN while not drastically increasing the number of layers in the network to improve training times and iteration.

My final model consisted of the following layers:

| Layer | Description |
|---|---|
| Input | 32x32x3 RGB image |
| Convolution 3x3 | 1x1 stride, valid padding, depth 12 outputs 30x30x12 |
| RELU | |
| Max pooling | 2x2 stride, outputs 15x15x12 |
| Convolution 4x4 | 1x1 stride, valid padding, depth 24 outputs 12x12x24 |
| RELU | |
| Max pooling | 2x2 stride, outputs 6x6x24 |
| Convolution 4x4 | 1x1 stride, valid padding, depth 48 outputs 3x3x48 |
| RELU | |
| Flatten | input 3x3x48, output 432 |
| Fully connected | input 432, output 216 |
| Fully connected | input 216, output 108 |
| Fully connected | input 108, output 43 |
| Softmax | Calculate Softmax probabiliy distribution of 43 outputs |

**Model Training:**

Output of the Softmax probability distribution of all 43 possible outputs is then compared to the training data labels using cross entropy (multi-nomial logistic classification). The average cross entropy (loss) is then calculated across the entire training set. Gradient descent is implemented using the AdamOptimizer function provided in TensorFlow in order to iterate CNN weights using a training rate of 0.0005. The batch size chosen for the training set was 128 examples, with a total Epoch number of 0.0005. Validation of the training data set was conducted using multi-nomial logistic classification as described above in order to determine the accuracy of the CNN over this separate data set, and to help determine accuracy increase during over each Epoch.

**Solution Approach:**

In order to achieve a validation accuracy of >0.93, a trial and error approach was used in construction of the CNN architecture from the baseline LeNet architecture. Initially, I implemented a CNN architecture which removed max pooling layers completely, and maximized depth through 5 separate Convolution-RELU layers achieving a final convolution layer depth of 120. However, while depth was drastically increased, the size of the flatten layer was also very large, requiring a large reduction in output size in each subsequent fully connected layer to achieve the final output size of size of 43. Furthermore, due to the large size of the CNN, the training time required to maximize accuracy was too long to be useful (~several hours), and in the end did not end up achieving an accuracy above 0.85.

I determined that due to the final depth of the convolution layers, data/information in the CNN was likely being squeezed out within the fully connected layers, as was required to reduce the data to the required output size. I chose to then reduce the final convolution layer output depth to be close to the required 43 output count, while making sure to only gradually increase layer depth with each convolution, such that information volume would be maintained. I also decided to implement re-implement max pooling between each convolution in order to help improve training time and reduce the final flatten layer size.

My final model results were:

- training set accuracy of 0.945
- validation set accuracy of 0.945

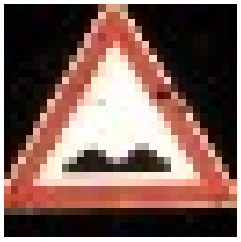
- test set accuracy of 0.945

## Test a Model on New Images

### Acquiring New Images:

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

"Speed limit (30km/h)", "Bumpy road", "General caution",  "No entry", "Wild animals crossing"



I chose these particular images to test the CNN, as they all have similar features which may make distinguishing each sign difficult.  Each sign has a red color implemented on the outer region of the sign, with varying black or white shapes on the inner portion.  Additionally, the background against which the sign is located varies drastically, further increasing difficulty for the CNN.  Furthermore, images 2, 3, and 5 (Bumpy Road, General Caution, Wild Animals Crossing) are especially similar to one another, sharing the same triangular shape, red outline, and dark color symbols on the against a white inner region.  The strength of the CNN will be tested in its ability to classify each sign based on these varying symbols.

### Performance on New Images:

The CNN predictions for the external images above were very accurate.  Each image was classified correctly, giving a 0.80 accuracy over the 5 image set.  This accuracy is consistent with the test dataset accuracy of 0.945, given the large discrepancy in dataset size.

Here are the results of the prediction:

| Image | Prediction |
|---|---|
| Speed Limit (30 km/h) | Speed Limit (30 km/h) |

| Image | Prediction |
|---|---|
| Bumpy Road | Speed Limit (30 km/h) |
| General Caution | General Caution |
| No Entry | No Entry |
| Wild Animals Crossing | Wild Animals Crossing |

### Model Certainty – Softmax Probabilities:

Softmax predictions for each external image were fairly accurate, with a low-end accuracy of ~0.99 not including the "Bumpy Road" image. Failure of the CNN to correctly predict the "Bumpy Road" image may be in part due to the limited training set examples (<500) for this particular label, which may bias predictions towards images with more examples (Speed Limit signs). Softmax predictions for each of the 5 external images is provided as follows:

#### Image - "General Caution"

| Probability | Prediction |
|---|---|
| 1.00 | General Caution |
| 4.76e-19 | Road Work |
| 1.46e-20 | Traffic Signals |
| 6.20e-22 | Bicycles Crossing |
| 3.48e-25 | Road Narrows on the Right |

#### Image - "Speed Limit (30km/h)"

| Probability | Prediction |
|---|---|
| 1.00 | Speed Limit (30km/h) |
| 4.42e-15 | Speed Limit (50km/h) |
| 1.97e-15 | Speed Limit (70km/h) |
| 1.62e-18 | Speed Limit (20km/h) |
| 2.92e-19 | Speed Limit (80km/h) |

#### Image - “Bumpy Road”

| Probability | Prediction |
|---|---|
| 0.99 | Speed Limit (30km/h) |
| 7.08e-4 | Bumpy Road |
| 2.59e-5 | No Entry |
| 2.35e-5 | Traffic Signals |
| 1.85e-6 | Speed Limit (20km/h) |

#### Image - “No Entry”

| Probability | Prediction |
|---|---|
| 0.99 | No Entry |
| 2.34e-5 | Turn Left Ahead |
| 1.33e-5 | Road Work |
| 8.30e-6 | Stop |
| 3.24e-6 | No Passing |

#### Image - “Wild Animals Crossing”

| Probability | Prediction |
|---|---|
| 0.99 | Wild Animals Crossing |
| 8.71e-4 | Road Narrows on the Right |
| 6.44e-4 | Slippery Road |
| 6.65e-5 | Double Curve |
| 1.58e-6 | Road Work |