

 writeup\_xh.md

# Traffic Sign Recognition

---

## Writeup Template

---

You can use this file as a template for your writeup if you want to submit it as a markdown file, but feel free to use some other method and submit a pdf if you prefer.

### 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

## Rubric Points

---

Here I will consider the [rubric points](#) individually and describe how I addressed each point in my implementation.

### Writeup / README

1. Provide a Writeup / README that includes all the rubric points and how you addressed each one. You can submit your writeup as markdown or pdf. You can use this template as a guide for writing the report. The submission includes the project code.

You're reading it! and here is a link to my [project code](#) on GitHub

### Data Set Summary & Exploration

1. Provide a basic summary of the data set and identify where in your code the summary was done. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.

The code for this step is contained in the code cell #4 of the IPython notebook.

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

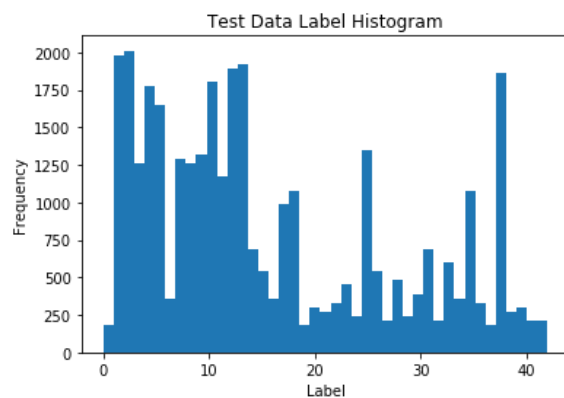
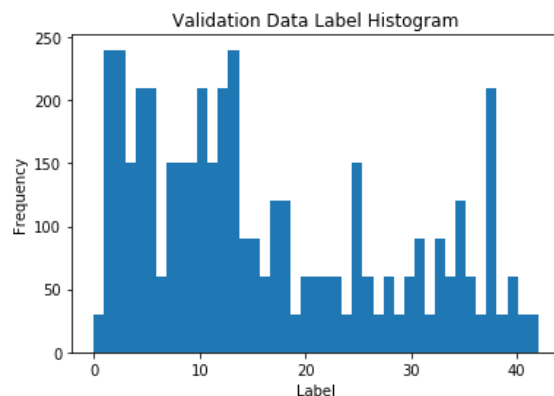
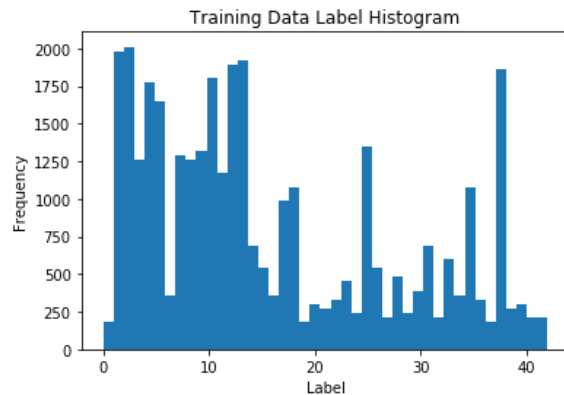
- The size of training set is 34799
- The size of validation set is 4410
- The size of test set is 12630
- The shape of a traffic sign image is 32x32x3 (RGB)
- The number of unique classes/labels in the data set is 43

This is just providing the raw metrics of the data set. The actual validation dataset I used is actually randomly splitted from the training data set. See the following section for more details.

2. Include an exploratory visualization of the dataset and identify where the code is in your code file.

The code for this step is contained in the code cell #7-9 of the IPython notebook.

Here is an exploratory visualization of the data set. I generated the histograms of the training, validation, and test sets:



It is quite obvious that our data sets are not evenly distributed. This raises a questions whether this distribution matches roughly what we expect in the real world or it is introducing artificial biasing due to data colleiton.

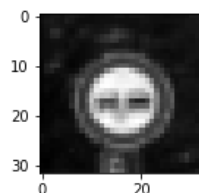
## Design and Test a Model Architecture

1. Describe how, and identify where in your code, 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.

The code for this step is contained in the code cell 11-13 of the IPython notebook.

I experimented with grayscaling and normalizaion. It seems that grayscaling did help to improve the accuracy in training, while normalization has very marginal effect. Therefore I only kept grayscaling and commented out normalization.

Here is an example of a traffic sign image before and after grayscaling.



2. Describe how, and identify where in your code, you set up training, validation and testing data. How much data was in each set? Explain what techniques were used to split the data into these sets. (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, identify where in your code, and provide example images of the additional data)

The code for splitting the data into training and validation sets is contained in the code cell 15 of the IPython notebook.

To cross validate my model, I randomly split the training data into a training set and validation set. I did this by using `train_test_split` from `sklearn.model_selection`

My final training set had 27839 number of images. My validation set and test set had 6960 and 12630 number of images.

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 cell #19 of the ipython notebook.

My final model consisted of the following layers:

| Layer           | Description                                 |
|-----------------|---|
| Input           | 32x32x1 grayscale image                     |
| Convolution 5x5 | 1x1 stride, valid padding, outputs 28x28x6  |
| RELU            |   |
| Max pooling     | 2x2 stride, outputs 14x14x6                 |
| Convolution 5x5 | 1x1 stride, valid padding, outputs 10x10x16 |
| RELU            |   |
| Max pooling     | 2x2 stride, outputs 5x5x16                  |
| Flatten         | Outputs 400                                 |
| Fully connected | Outputs 120                                 |
| Fully connected | Outputs 84                                  |
| Fully connected | Outputs 43                                  |
| Softmax         |   |

4. 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 cell 22-27 of the ipython notebook.

To train the model, I used the AdamOptimizer, batch size of 256, learning rate of 0.001 and 35 epochs. I experimented with various batch size, learning rate and number of epochs and this set is among the best performing ones without very large epoch numbers. I also tried different sigma for initializing the weights and determined 0.05 is a relatively good number. The training accuracy reaches above 99% within 35 epochs.

5. Describe the approach taken for finding a solution. 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.

The code for calculating the accuracy of the model is located in the cell #28 of the lpython notebook.

My final model results were:

- training set accuracy of 99.7%
- validation set accuracy of 98.4%
- test set accuracy of 91.3%

I chose LeNet because it seems to perform well enough for this task. The training and validation accuracy is very high while the test accuracy is still acceptable (above 91%).

I experimented with various batch size, learning rate and number of epochs and this set is among the best performing ones without very large epoch numbers. I also tried different sigma for initializing the weights and determined 0.05 is a relatively good number.

I also added dropouts to see its effect. Either adding dropout layers after maxpooling of the convolutional layers or after the fully connected layers. Unfortunately it did not help with improving accuracy. In both cases, the training, validation, and test accuracies all dropped with some extent.

The final training and validation accuracies are both high and are pretty close, within about 1% difference, which indicates good fitting.

## 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 ...

2. Discuss the model's predictions on these new traffic signs and compare the results to predicting on the test set. Identify where in your code predictions were made. 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).

The code for making predictions on my final model is located in the cell 30-33 of the lpython notebook.

Here are the results of the prediction:

| Image   | Grundtruth    | Prediction    |
|---|---------------|---------------|
|  | Pedestrians   | Pedestrians   |
|  | Priority road | Priority road |

| Image   | Grundtruth       | Prediction       |
|---|------------------|------------------|
|  | Speed limit 30   | Speed limit 30   |
|  | Turn right ahead | Turn right ahead |
|  | Speed limit 30   | Speed limit 50   |

The model was able to correctly guess 5 of the 5 traffic signs, which gives an accuracy of 100%. This compares favorably to the accuracy on the test set of 91.3%.





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 cell #33 of the lpython notebook.


The model is highly certain about the first three images(100% probabliity). The sign of turn right ahead is also predicted with a high probability of 99.6%. Only the speed limit of 50 has a lower prediction probability of 93.1%, but still above the test accuracy.

| Image   | Probability | Prediction       |
|---|-------------|------------------|
|    | 100.0%      | Pedestrians      |
|   | 100.0%      | Priority road    |
|  | 100.0%      | Speed limit 30   |
|  | 99.6%       | Turn right ahead |
|  | 93.1%       | Speed limit 50   |

The table below illustrates the top 5 softmax outputs. The #2-#5 guesses of each sign are all with very low probability, close to zero for all other guesses for the first four images. Only Speed limit 30km/h had 6.92% probably for the last image (Speed limit 50 km/h).

| Image   | Prediction       | 1                | 2                    | 3                                     | 4                     | 5  |
|---|------------------|------------------|----------------------|---------------------------------------|-----------------------|--|
|  | Pedestrians      | Pedestrians      | General caution      | Roundabout mandatory                  | Traffic signals       | Right-of-way at the next intersection        |
|  | Priority road    | Priority road    | Roundabout mandatory | Right-of-way at the next intersection | Speed limit (100km/h) | No passing for vehicles over 3.5 metric tons |
|  | Speed limit 30   | Speed limit 30   | Yield                | Speed limit (70km/h)                  | Speed limit (50km/h)  | Keep left                                    |
|  | Turn right ahead | Turn right ahead | Traffic signals      | General caution                       | Speed limit (70km/h)  | Keep left                                    |

2/28/2017writeup\_xh.md - Grip

| Image   | Prediction        | 1                 | 2                       | 3                       | 4                          | 5                    |
|---|-------------------|-------------------|-------------------------|-------------------------|----------------------------|----------------------|
|  | Speed limit<br>50 | Speed limit<br>50 | Speed limit<br>(30km/h) | Speed limit<br>(80km/h) | Speed<br>limit<br>(20km/h) | Speed limit (70km/h) |