1. Traffic Sign Recognition

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

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

Data Set Summary & Exploration

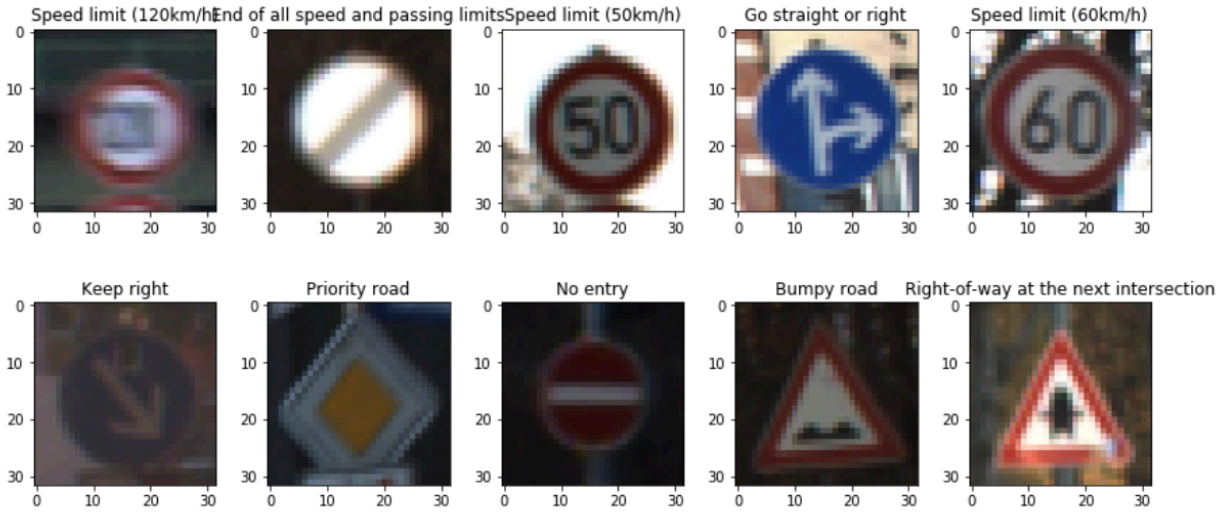
1. Provide a basic summary of the data set. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.

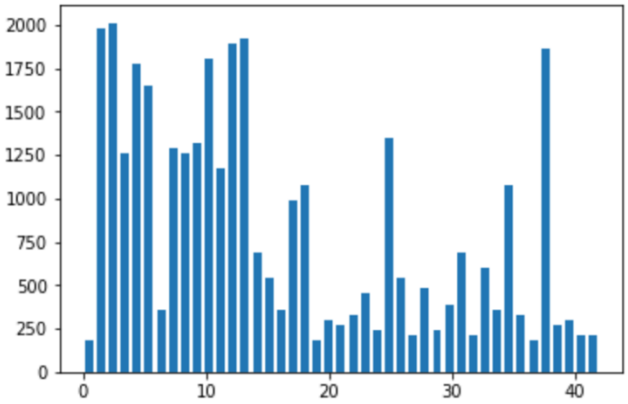
I used the numpy library to calculate summary statistics of the traffic signs data set:

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

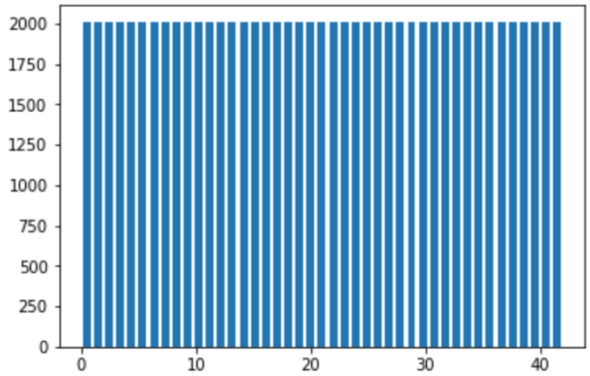
2. Include an exploratory visualization of the dataset.

Here are two images that visualize the dataset. First is ten labeled example images with their labels, and second is a histogram displaying how well each class is trained. Some classes are better trained than others and this can be improved by augmenting the dataset.





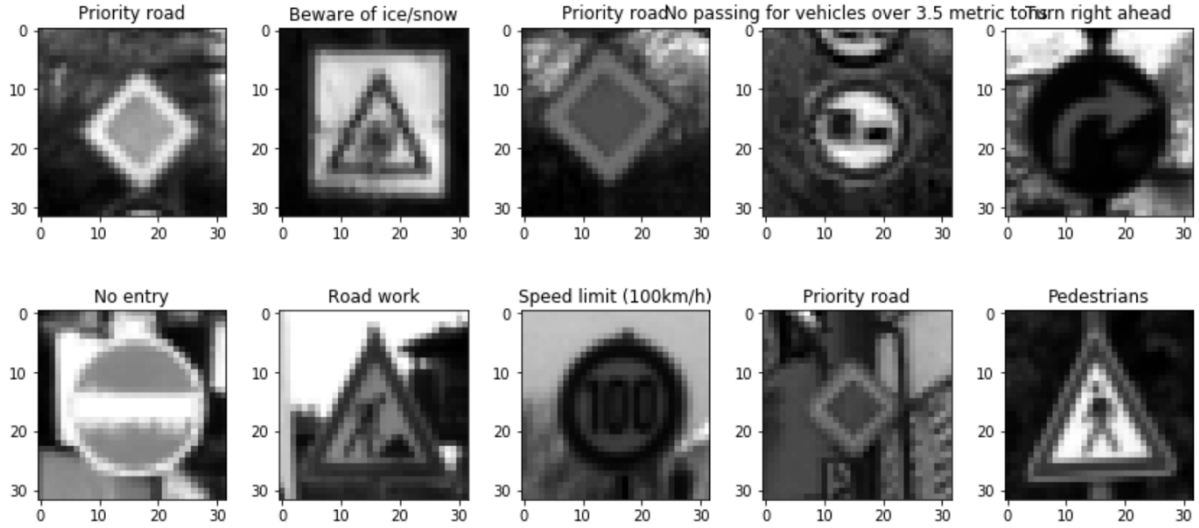
Edit: I made two changes to the dataset—First, I made the dataset equal to the highest trained class which resulted in the following histogram. Second, I rotated the images.



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.)

The first step was to grayscale the image and then normalize it. The image below is the output of processed image. Though the lessons suggested I should subtract by mean value and divide by standard deviation of the mean value, I chose to use the cv2 library since it has a function cv2.normalize(). Similarly, I used cv2 to convert from RGB to GRAY as well.



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	Description
Input	32x32x1 Grayscale image
Convolution 1x1	1x1 stride, Valid padding, outputs 28x28x20

RELU	
Maxpool	Outputs 14x14x20, 2x2 strides
Input	14x14x20 RGB image
Convolution 1x1	1x1 strides, Valid Padding, outputs 10x10x40
RELU	
Maxpool	2x2 strides, Outputs 5x5x40
Flatten	
Fully Connected	Input 1000, output 500
RELU	
Fully Connected	Input 500, Output 120
RELU	
Fully Connected	Input 120, Output 84
RELU	
Fully Connected	Input 84, Output 43
Softmax	Output 43 logits

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 the following parameters:

Learning rate: 0.0005. I chose the learning rate of 0.002 as it was already used during the lessons. However, I changed the learning rate after a bit of research that concluded that a learning rate between 0.0005 and 0.001 would work the best for Adam optimizer. This was mostly chosen through trial and error.

Optimizer: Adam Optimizer. I looked up different optimizers and it turned out that Adam Optimizer learned fastest and better than others. Since we already learned using this optimizer, I went with it for the project as well.

Batch size: 128. I tried other batch sizes but this seemed most suitable and efficient.

Epochs: 200. I changed Epochs quite often depending how much GPU time I wanted to use. Since I was able to get minimum accuracy from the earliest trainings, I had kept the Epochs at 20 until I was ready to test a complete code.

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:

- training set accuracy of ? ***100% (max)***
- validation set accuracy of ? ***97% (max)***
- test set accuracy of ? ***95.234%***

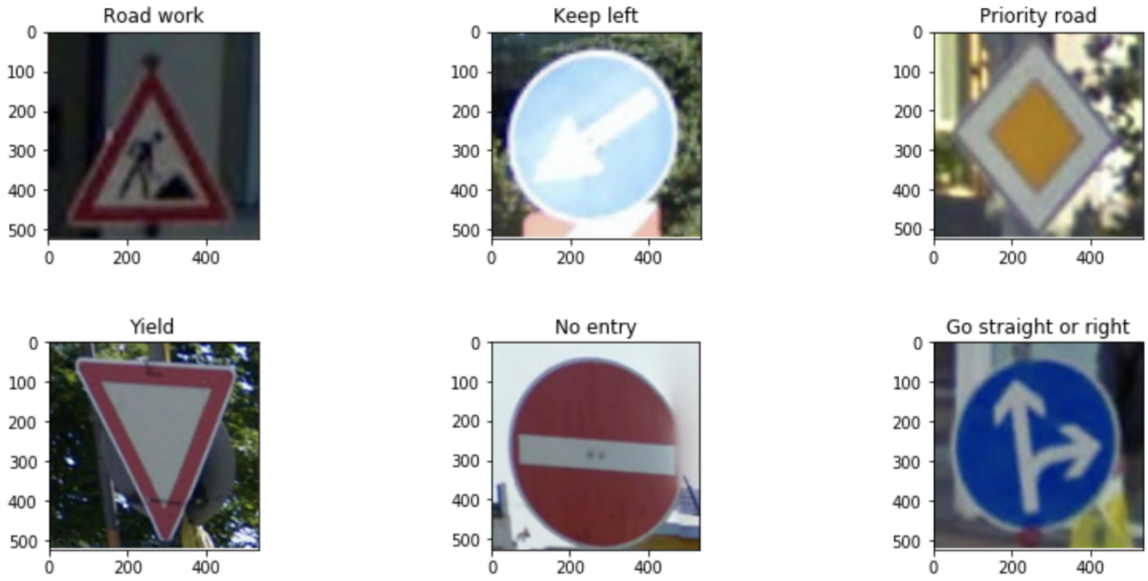
If a popular approach was chosen:

> The first architecture I tried was LeNet since we already had used it for course exercises and is pretty well-known.
>
> The initial architecture had 2 convolution layers, 2 pooling layers, and 3 fully connected layers. It had the depth of 6 in first layers and 16 in the second layers. There wasn't necessarily any problem with this architecture but it seemed too simple so I made some changes.
>
> I played around with the architecture in order to determine what gives me the best results after training. I added depth to the initial layer, making the depth 20, and as I decreased the matrix dimensions, I increased depth to 40, totaling to 1000 on input for fully connected layer. I added an additional fully connected as I brought down the input from 1000 to 43 (number of our classes). I didn't add a dropout layer because I didn't think it was necessary. I believe the architecture was over-fitting since my validation accuracy was slight lower than training accuracy.
>
> During the trial and error, I also added a second LeNet function to process the images twice but it didn't seem to work for me and worsened the training accuracy so I went back to using single LeNet.

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.

The images I chose were:

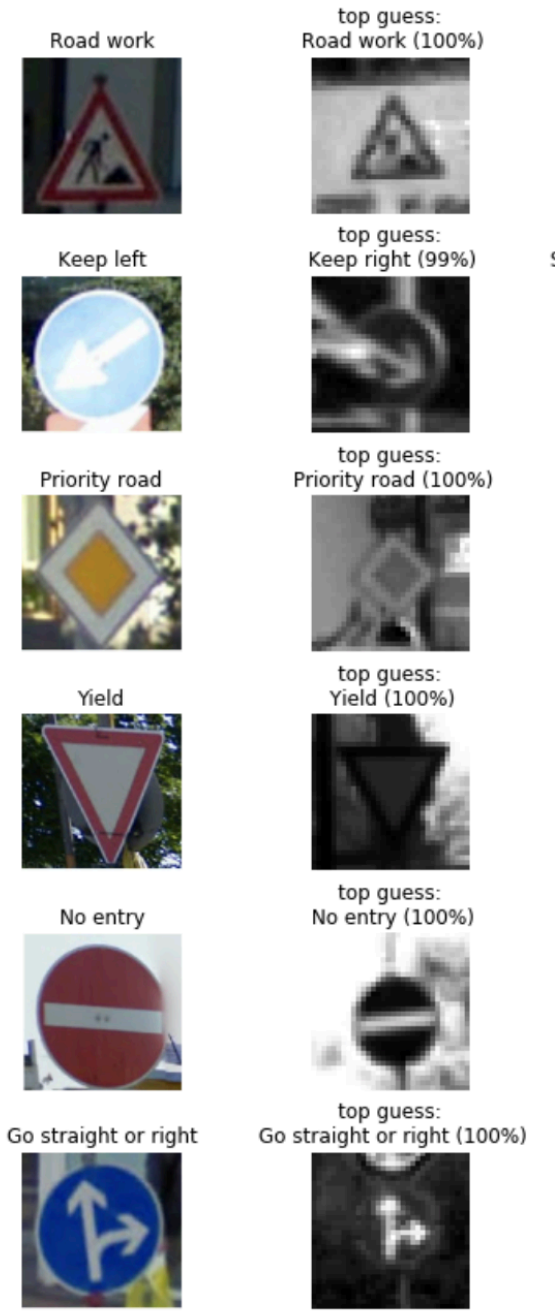


The most difficult sign among these has been keep left sign. Perhaps because of the brightness of the image, or perhaps it is the arrow, but it's a hard one to determine. I haven't tried rotating images yet but I believe that should help the case.

Edit: I added rotation to image processing and while it improved the overall accuracy, it didn't help with the 'Keep Left' sign.

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).

As we can see below, the model was able to predict 5 out of 6 given signs accurately, which is about 83% of accuracy. The 'keep left' sign was predicted as 'keep right' and for that, I plan to add a rotation processing of image and see if it helps the training.



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 model was able to predict the signs relatively accurate. I have 6 instead of 5 new signs so I got 83.3% accuracy.

Probability	Prediction	Original
1.0	Yield	Yield
1.0	Road work	Road Work
1.0	Priority Road	Priority Road
1.0	Go Straight or Right	Go Straight or Right
1.0	No Entry	No Entry
0.9889	Keep Right	Keep Left

(I did not do Step 4)