Traffic Sign Recognition

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 second code cell of the IPython notebook.

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

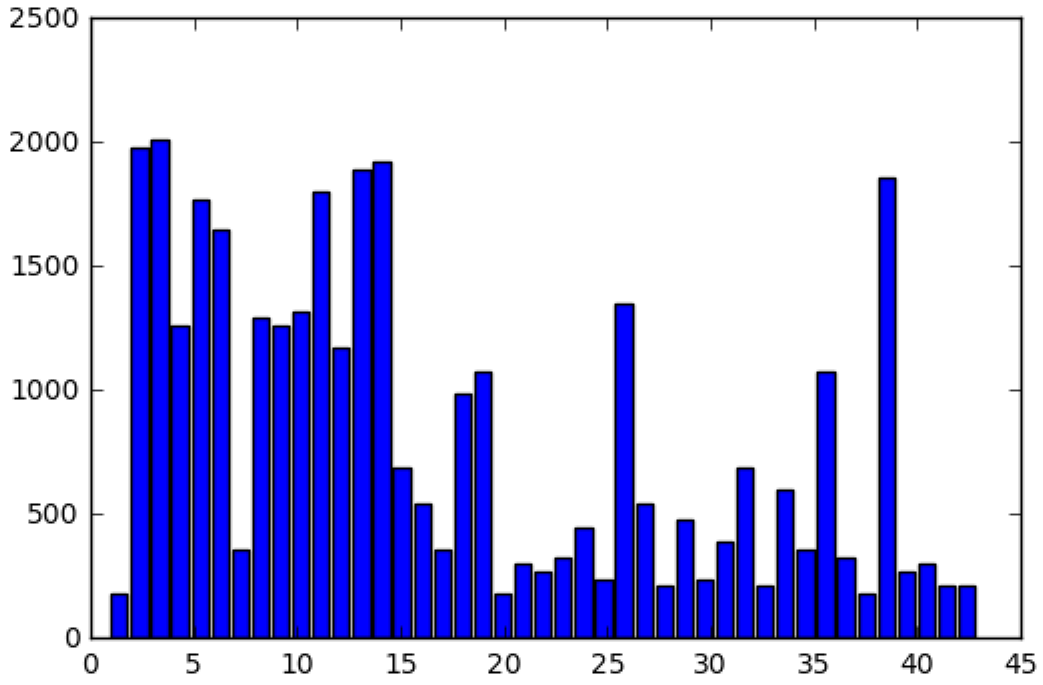
- The size of training set is 34799
- The size of test set is 12630
- The shape of a traffic sign image is (32, 32, 3)
- The number of unique classes/labels in the data set is 43

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 third code cell of the IPython notebook.

Here is an exploratory visualization of the data set. It is a random sample of the data. In this case it is of class 12.

A histogram showing the amount of data for each label:



As we can see there are a lot of data for some classes, but there isn't a lot of data for some other classes, like label 1 and 20. This makes the data distribution imbalanced and can cause the model to overfit to classes with more data.

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 fourth code cell of the IPython notebook.

I shuffled the data so that it's in a random order, because the model might infer the ordering of the data might have an effect on the result.

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 first code cell 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 loading the pickle files and assign them to the corresponding data sets.

My final training set had 34799 number of images. My test set had 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 fifth cell of the ipython notebook.

My final model consisted of the following layers:

Layer	Description
Input	32x32x3 RGB image
Convolution 3x3	1x1 stride, valid padding, outputs 28x28x6
RELU	
Max pooling	2x2 stride, outputs 14x14x6
Convolution 3x3	1x1 stride, valid padding, outputs 10x10x16
RELU	
Max pooling	2x2 stride, outputs 5x5x16
Flatten	
Fully Connected	Input is400, out is 120
RELU	
Dropout	Keep half for training
Fully Connected	Input is 120, output is 84

RELU	
Fully Connected	Input is 84, output is 43

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 eighth cell of the ipython notebook.

To train the model, I used an Adam Optimizer with a learning rate of 0.0005 for 100 epochs. The batch size is 128.

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 ninth cell of the Ipython notebook.

My final model results were:

- validation set accuracy of 0.937
- test set accuracy of 0.929

If a well known architecture was chosen:

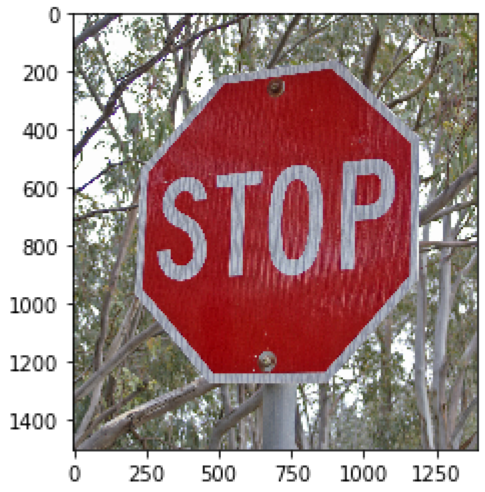
- What architecture was chosen?
  - I went with LeNet from the class
- Why did you believe it would be relevant to the traffic sign application?
  - LeNet was designed for classifying images, and I started out using it for this project and it has turned out decent with minimum amount of adjustments
- How does the final model's accuracy on the training, validation and test set provide evidence that the model is working well?
  - I only added a dropout layer and lowered the learning rate, and upped the epochs compensate for the lower learning rate, and it increased the accuracy from the mid 0.8s to low 0.9s

Test a Model on New Images

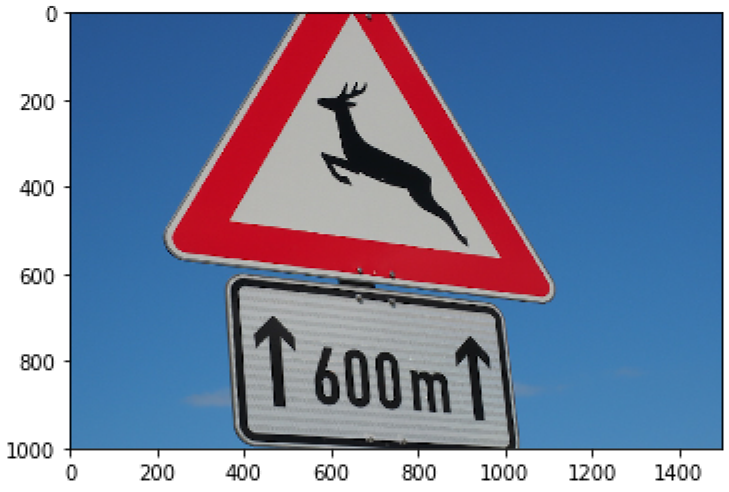
1. Choose five German traffic signs found on the web and provide them in the report. For each image, discuss what quality or qualities might be difficult to classify.

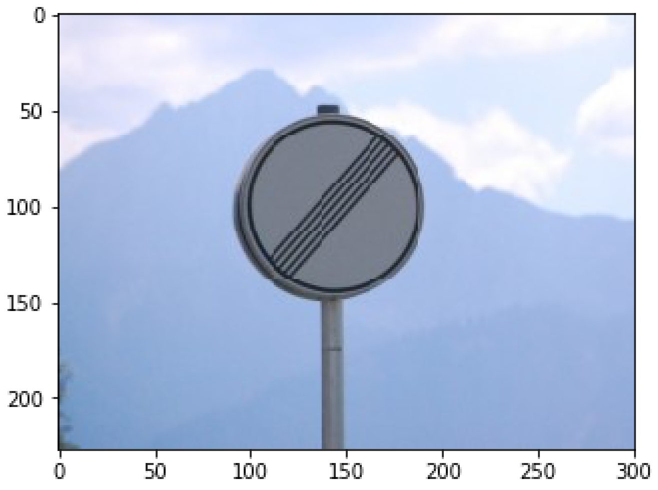
Here are five German traffic signs that I found on the web:

120
0
50
100
150
200
0
100
200
300
400


STOP
0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250

0
25
50
75
100
125
150
175
200
225
0
50
100
150
200
250
300





1. The actual sign is only half of the picture. The model has to figure out what the blur on the left side is and how that affects the prediction. Also the sign is cut off leaving only part of the sign.
2. There are some lines on the sign.
3. This sign should not be difficult to classify.
4. This sign has another sign accompanying it, which could confuse the model. Also the top is cut off.
5. The background does not have a big contrast to the sign.

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 fourteenth cell of the Ipython notebook.

Here are the results of the prediction:

Image	Prediction
Speed limit (120km/h)	Speed limit (30km/h)
Stop	Stop
Road work	Road work
Wild animals crossing	Keep right
End of all speed and passing limits	Yield

The model was able to correctly guess 2 of the 5 traffic signs, which gives an accuracy of 40%. This compares unfavorably to the accuracy on the test set of 0.929

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 last cell of the Ipython notebook.

For the first image, the model is relatively unsure that this is a speed limit of 30km/h (probability of 37.1055603), and the image does contain a speed limit sign. The top five soft max probabilities were

Probability	Prediction
37.1055603	Speed limit (30km/h)
32.53974152	Roundabout mandatory
22.61756134	Priority road
14.19962883	Turn right ahead
12.2980051	Speed limit (70km/h)

For the second image, the model is relatively sure that this is a stop sign (probability of 47.12952423), and the image does contain a stop sign. The top five soft max probabilities were

Probability	Prediction
47.12952423	Stop
24.12884331	No entry
9.79175663	Bicycles crossing
7.2927475	Priority road
5.89428186	General caution

For the third image, the model is relatively sure that this is a Road work sign (probability of 48.69383621), and the image does contain a road work sign. The top five soft max probabilities were

Probability	Prediction
48.69383621	Road work
17.95225143	Beware of ice/snow
15.41103172	Road narrows on the right
15.24446297	Speed limit (30km/h)
14.67629623	Bicycles crossing

For the fourth image, the model has no idea what this is (probability of 1.53398287), and the image technically does contain two signs. The top five soft max probabilities were

Probability	Prediction

1.53398287	Keep right
1.12710428	Priority road
0.92617548	Dangerous curve to the right
0.67257738	Speed limit (60km/h)
0.6208483	Speed limit (80km/h)

For the fifth image, the model is relatively unsure that this is a Yield sign (probability of 18.73088455), and the image does contain a sign for End of all speed and passing limits. The top five soft max probabilities were

Probability	Prediction
18.73088455	Yield
18.32155991	Turn left ahead
14.25983906	Dangerous curve to the right
8.78882504	End of no passing
8.34080887	Children crossing