Traffic Sign Recognition

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

Data Set Summary & Exploration

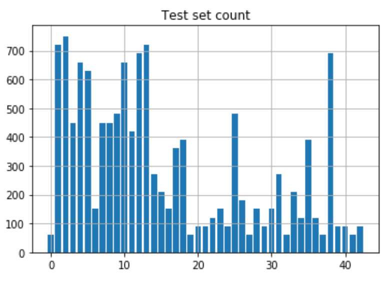
1. Provide a basic summary of the data set. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.

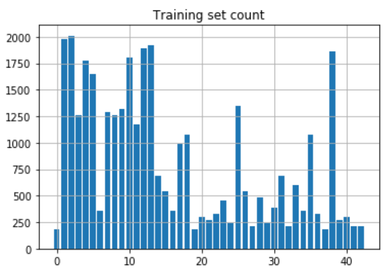
I used the numpy library to calculate summary statistics of the traffic signs data set:

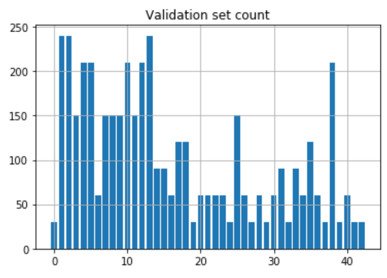
- The size of training set is 34799
- The size of the validation set is 4410
- The size of test set is 12630
- The shape of a traffic sign image is (32,32,3)
- The number of unique classes/labels in the data set is 43

2. Include an exploratory visualization of the dataset.

The following code generates a frequency histogram of the training, validation and test dataset. This helps in visualising the distribution of the entire dataset among individual classes and thus detect potential bias towards a particular class. This can be rectified by augmenting the dataset by generating more examples for classes that do not have a good enough representation. Since the distribution is not uniform across training,test and validation, data augmentation is required.







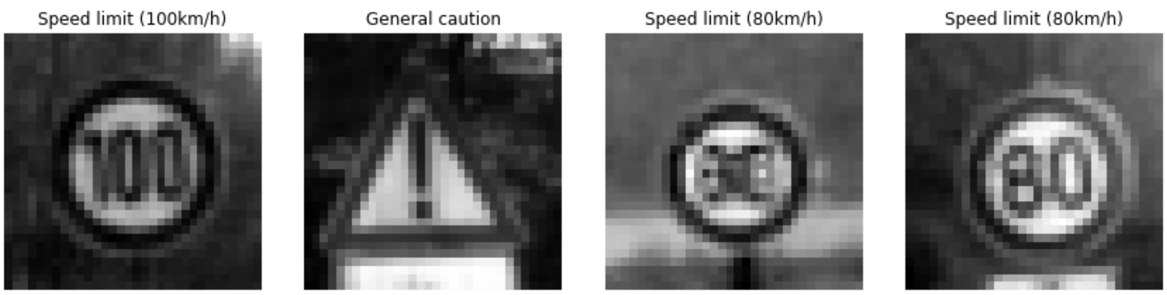
Design and Test a Model Architecture

1. Data preprocessing

Conversion to grayscale

Firstly, the images were converted to grayscale because as mentioned in this paper, converting images to grayscale greatly improves the accuracy of my model

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



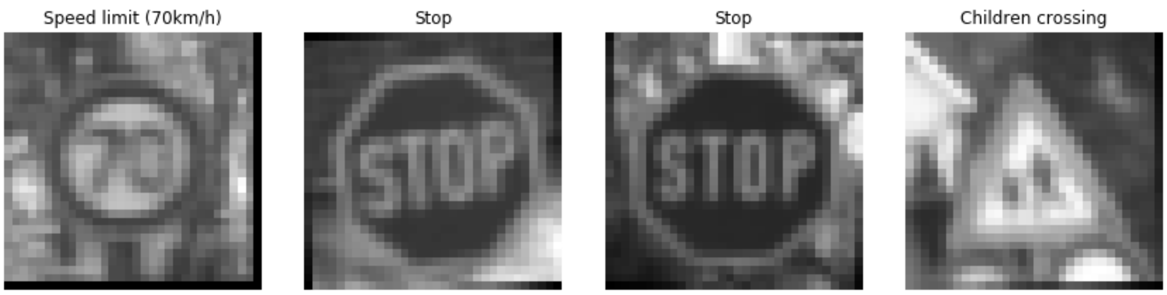
Data Augmentation

After grayscaling, the images are augmented(using rotation, scaling and perspective transform) in such a way that each class has a representation of at least 3000 images. This helps in removing any potential bias and also helps in generating a larger training and validation set. In the last step, the training and validation set is merged and re-split again to incorporate distorted images into the validation set.

Finally, the split of dataset is as follows:
Training set: 119106
Validation set: 12630
Test set: 29777

Normalization

As a last step, I normalized the image data such that the dataset has zero mean and unit standard deviation across all features. This makes the model more robust and immune to biases due to certain features.



2. Final Model Architecture

My final model consisted of the following layers:

Layer	Description
Input	32x32x1 Grayscale 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
Fully connected	Input: 400(5x5x16) Output: 120
RELU	
Dropout	Keep_probability = 0.5
Fully connected	Output: 84
RELU	
Dropout	Keep_probability = 0.5
logits	Output: 43

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 Adam Optimizer as it is the most robust optimizer and is recommended in all discussion formus. As it uses momentum, it helps in reaching the minimum faster and avoids deviation of the loss function. The batch size is taken as 256 and number of epochs is 47. The learning rate is 0.001. The parameters were chosen based on intuition and trial and error.

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,

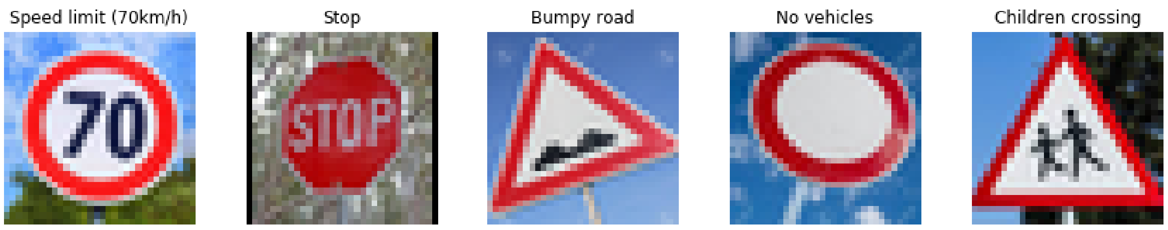
validation and test sets and where in the code these were calculated. Your approach may have been an iterative process, in which case, outline the steps you took to get to the final solution and why you chose those steps. Perhaps your solution involved an already well known implementation or architecture. In this case, discuss why you think the architecture is suitable for the current problem.

My final model had a set accuracy of about 95.5% and a test set accuracy of 90.1%. To implement this, firstly, I directly used the LeNet architecture from the exercises, with a few modifications such as the input channel had three features instead of one and the final layer had 43 classes instead of 10. This gave me an accuracy of about 89%. The main problem was that in the final layers of the architecture, a lot of neurons were overfitting. Thus to tackle this problem, I introduced 2 dropout layers to after each activation of the fully connected layers to prevent overfitting. Finally, the hyperparameters were tuned using intuition and trial and error.

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 stop sign is typically difficult to identify as there is a lot of background noise and the image has been cropped. The 3rd and 4th image are also at an angle(have been rotated) and there aren't many examples of the rotated images. So the model may not fare well.

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 a part of the preprocessing, I converted the images to grayscale and normalized them as done with the images previously. After that the images looked like this:
Here are the results of the prediction:

The predictions were the following:

Image	**Prediction**
70 km/h	Stop sign
Stop Sign	Stop sign
Bumpy Road	Children crossing
No vehicles	Beware of ice/snow
Children crossing	30 km/h

The model was able to correctly guess 1 of the 5 traffic signs, which gives an accuracy of 20%. This result is inconsistent with the test accuracy, which is about 90%.

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.

The prediction with the highest probability is the one that is predicted by the model. The top five softmax probabilities for each image were:

Probability	**Prediction**
.58	30km/h
.21	50km/h
.12	No vehicles
0.02	100 km/h
0.02	Stop sign

Probability	**Prediction**
1.0	Stop sign
0	50 km/h
0	80 km/h
0	Priority road
0	30 km/h

Probability	Prediction
0.51	No passing
0.22	60 km/h
0.06	End of No passing
0.05	Priority road
0.05	Vehicles over 3.5 metric tons prohibited

Probability	Prediction
0.56	Pedestrians
.36	Road narrows on the right
.05	Children crossing
.01	Right-of-way at the next intersection
.01	Bicycles crossing

Probability	Prediction
.77	70 km/h
.2	Ahead Only
0.01	Roundabout mandatory
0.01	20 km/h
0.01	Go straight or left