Traffic Sign Classifier

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

Following gives a summary of basic dataset statistics. It is done in the second code cell of the attached Ipython Notebook. Numpy is used to produce some of the results.

Number of training examples = 34799
Number of validation examples = 4410
Number of testing examples = 12630
Image data shape = (32, 32, 3)
Number of classes = 43

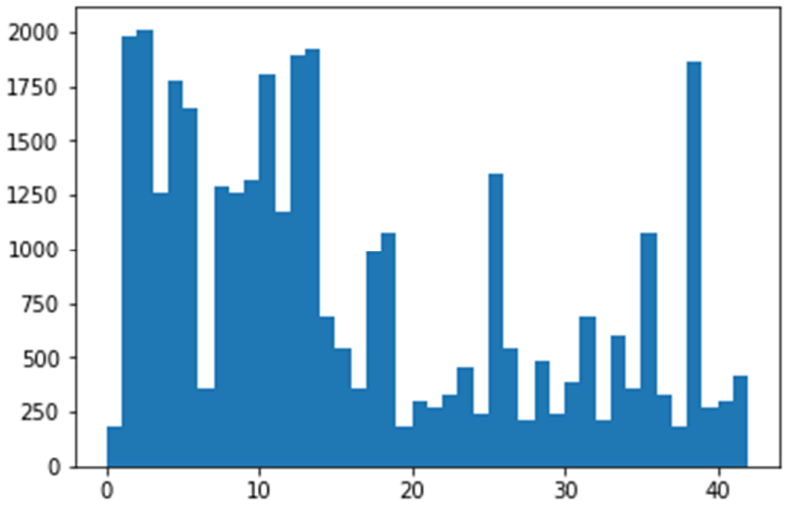
From the data it can be deduced that:

1- There are ~35000 training images of size 32,32,3
2- There are 43 different types of traffic signs to classify into

Exploratory Visualization

Here is an exploratory visualization of the data set. The picture below gives one example each of all the different traffic signs available.

A histogram is generated to get an idea of how many images are present for each class in the training dataset. It is done in the code cell 3 of the notebook.



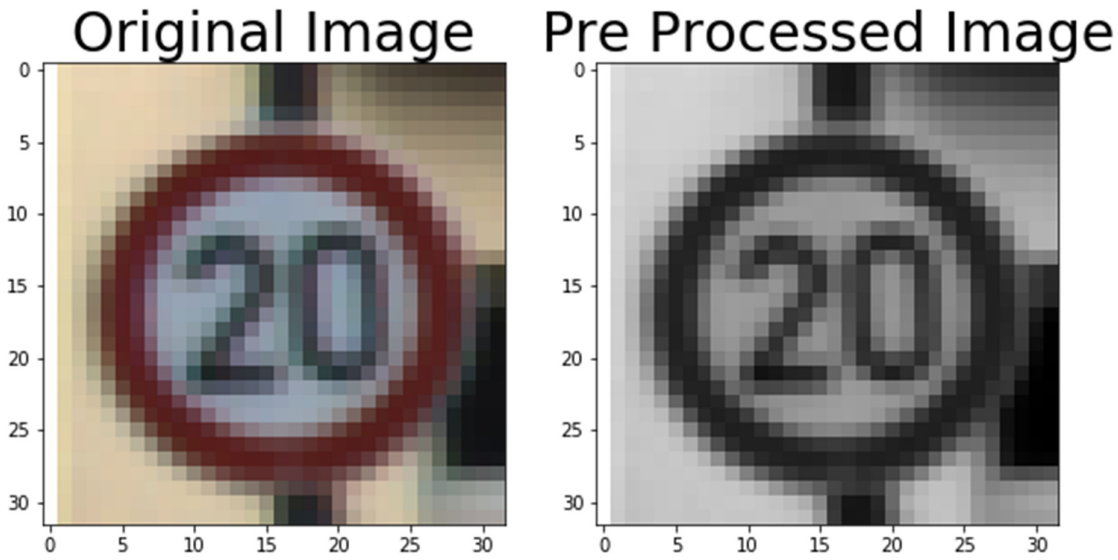
As you can see the dataset is not balanced there are a few classes with much more examples than the others.

Design and Test a Model Architecture

Preprocessing

As a first step, I decided to convert the images to grayscale because looking at the images it doesn't seem that colour information is of much use and only going to be extra work for the classifier. It is better to eliminate that. I also normalised the images so that the data has mean zero and equal variance

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



Data Augmentation:

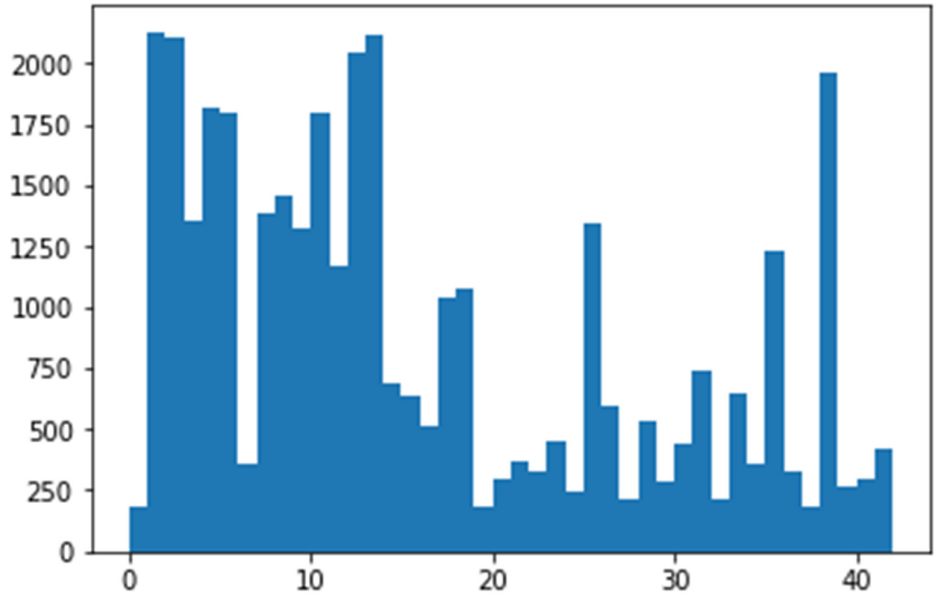
I tried first pass with the original dataset and I was getting stuck at 92% accuracy. To improve it, I tried to get more training data by augmenting the dataset. In order to augment the dataset, a random brightness, rotation, translation and shear is applied to the image.

I used guidelines from Vivek Yadav's post on medium for the data augmentation. 50 additional images are generated for each class and added to the dataset. This gave me 1% extra accuracy on validation set. Example of data augmentation:

Number of training examples after augmentation = 36949

The shape of images in the dataset (32, 32)

The histogram after the augmentation



Final model architecture:

I used Lenet as my final architecture. The images are converted to the grayscale so they are fed into the network as 32x32x1 size. The final layer gives the output as 43 classes.
The network consists of 2 convolutional layers(5x5) with RELU activation (to introduce non linearity) and max pooling followed by 3 fully connected layers. I also used a dropout layer in between to improve the accuracy.

Input	32x32x1 Grayscale image
Convolution 5x5	1x1 stride, valid padding, Output = 28x28x6.
RELU	
Max Pooling	2x2 strides, Output = 14x14x6
Convolution 5x5	1x1 stride, valid padding, Output = 10x10x16..
RELU	
Max Pooling	2x2 strides, Output = 5x5x6
Flatten.	Input = 5x5x16. Output = 400
Fully Connected	Input = 400. Output = 120
RELU	
Dropout	0.5
Fully Connected	Input = 120. Output = 84
RELU	
Dropout	0.5
Fully Connected	Input = 84. Output =43

Network Parameters and Training

I first started with Lenet as it is one of the oldest known and efficient architecture. It has successfully worked for various image classifer projects. Alos at the very first pass, it gave me about 90% accuracy. So I thought it is a good starting point. After which I preprocessed the data, augmented the dataset with more example pictures as explained above and used a dropout layer (on recommendation from the course forums) which improved the validation accuracy to 95.6%. The dropout is used as a tool to reduce overfitting.

The following parameters are used:
Type of optimizer: Adam
learning rate= 0.001
EPOCHS = 50
BATCH_SIZE = 128

I did not change Type of optimizer and batch size. I played with different learning rates and epochs to arrive at the result. For me, the validation accuracy was not increasing much for after about 50 epochs so I stopped there. The learning rate was also iterated from 0.01 and 0.0001 I found that 0.001 works well for this.

This iterative procedure gave me the following results:

Validation set accuracy: 95.7%
Test set accuracy: 93.6%

The test set accuracy of 93.6% shows that the model works well on the images that it has never seen before and has no way to memorize. I applied the same pre-processing to validation and test set as well.

Test a Model on New Images

I found these 5 German dataset images on the internet:

They belong to following classes:

Sign Name	Class ID
Keep Right	**38**
Speed Limit 30 kmph	**1**
No Entry	**17**
Children Crossing	**28**
Stop	**14**

I chose this dataset as it has various shapes- circles vs triangle vs octagon and specifically these features:

- The children crossing sign has some peculiar shapes inside the triangle.
- STOP sign has words inside
- speed limit sign has numbers inside
- keep right has a new shape- an arrow
- No entry is similar to some other signs like No passing

The actual figures were also different size in the pictures and have different zoom. I had to crop a certain region for children crossing sign as the original picture has things written under it

Model predictions

The model predicted correct results for 3 out of 5 images

Test Set Accuracy = 0.600

Here are the predictions for the 5 images:

Traffic Sign	**Prediction**
Keep Right	**Keep Right**
Speed Limit 30 kmph	**Speed Limit 30 kmph**
No Entry	**17**
Children Crossing	**Children Crossing**
Stop	**14**

The model was able to correctly guess 3 of the 5 traffic signs, which gives an accuracy of 60%. This compares favourably to the accuracy on the test set of 93.7%. How certain the model is on every prediction and why the model failed on two images is discussed in next section.

Top 5 softmax values

Here I am going to discuss the top 5 softmax probabilities for every image. Every probability which is less than 0.00001 has been written as 0 as that less of a number doesnt mean much

For the first image, the model predicted with 99% probability for the Keep right and it was a right prediction

Image1	**Keep Right**
Probability	**Prediction**
0.99	Keep Right
0.04	No passing for vehicles over 3.5 metric tons
0	
0	
0	

For the second image, the model predicted incorrectly in favour of wild animals crossing. The next best guess was the speed limit 30 kmph which is the right guess. It is followed by a number of other speed limit guesses as they look pretty similar.

Image2	**Speed limit (30km/h**
Probability	**Prediction**
0.7	Wild animals crossing
0.27	Speed limit (30km/h
0.02	Speed limit (50km/h)
0.007	Speed limit (80km/h)
0.001	Keep Right

For the third image, model predicted correctly the no entry sign with a probability of 1

Image3	**No Entry**
Probability	**Prediction**
1	No entry
0	
0	
0	
0	

For fourth image, model predicted correctly the children crossing sign with a probability of 1

Image4	**Children crossing**
Probability	**Prediction**
1	Children crossing
0	
0	
0	
0	

For the fifth image, model failed miserably as none of the top 5 predictions were stop sign which is the image.

Image5	**Stop**
Probability	**Prediction**
0.8	Priority road
0.1	Speed limit (100km/h)
0.07	Roundabout mandatory
0.005	End of no passing by vehicles over 3.5 metric tons
0.004	No passing

So from the above test set, we can say that model is very certain when it is producing the right results. However there are instances where the model failed completely like in the case of STOP sign. From the forums, I came to know that the type of image create a large impact on model predictions even if the model is producing good results on validation and test set. I believe that's what happened on the Stop sign. I cropped that picture from a large picture.