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

Dataset summary and exploration

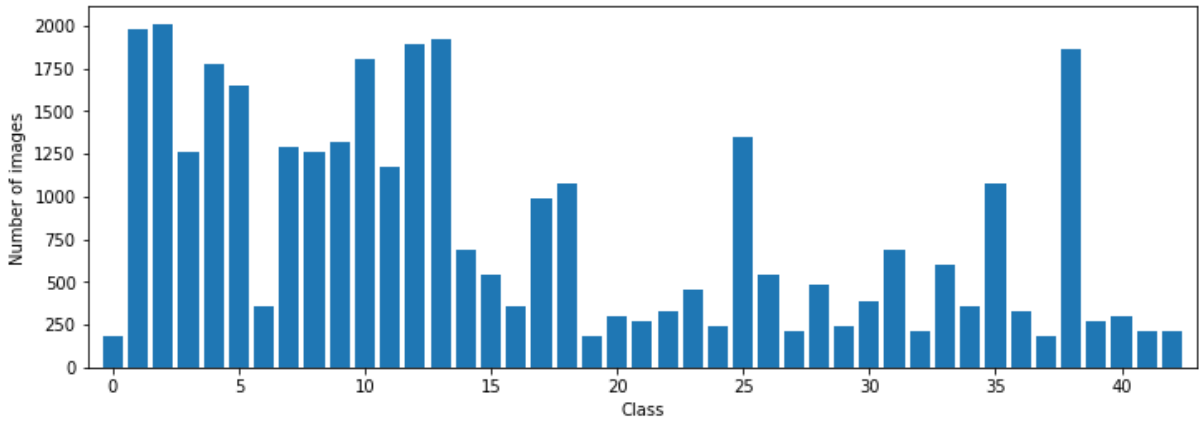
I used the numpy library to calculate summary statistics of the traffic signs data set:

- Number of training examples = 34799
- Number of testing examples = 12630
- Number of validation examples = 4410
- Image data shape = 32X 32, 3 channel RGB image
- Number of classes = 43

Exploratory visualization of the dataset

A few of the example images of different classes before image processing. As obvious from the images, there is a huge variation in the image contrast and brightness which could be fixed using some preprocessing before it can used to train the network.

The plot below shows the distribution dataset images and we can observe that the distribution is not uniform as some classes of the traffic signs have more examples than others.



I have used the pipeline described in “Traffic Sign Recognition with Multi-Scale Convolutional Networks” by Yann LeCun and Pierrre Sermanet. Here, the examples are converted into grayscale and the paper only considers the ‘Y’ channel. I perform the same operation on all the examples. The images were then normalized. I noticed that there was a lot of variation of brightness between the images of the same class. This was fixed by performing localized histogram equalization and mean normalization. A few examples for the converted images and histogram normalized images are shown below. The normalization helps immensely in low light images, for example image 4 (60 mph) sign is not visible very clearly but on the histogram normalized image 4 is very clear. I did not have to augment the data to obtain the a validation accuracy of over 97% and to prevent over-fitting, I used dropout factor of 0.8 at each layer.

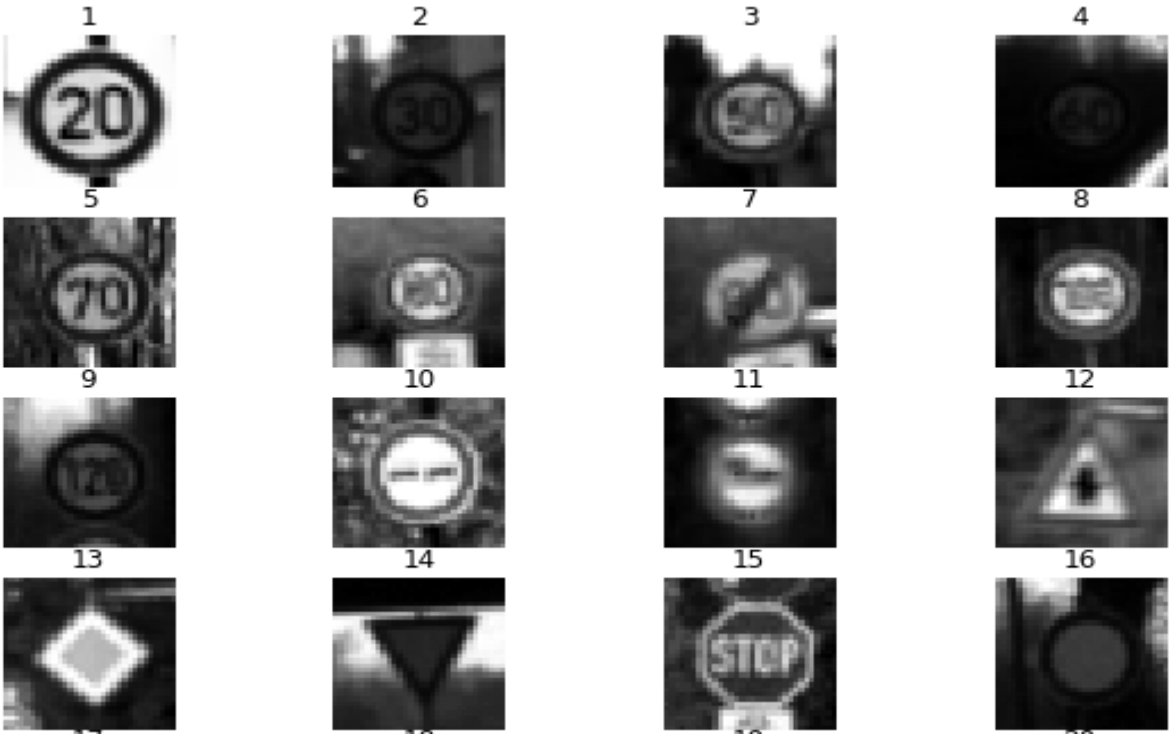


Figure: RGB to gray conversion

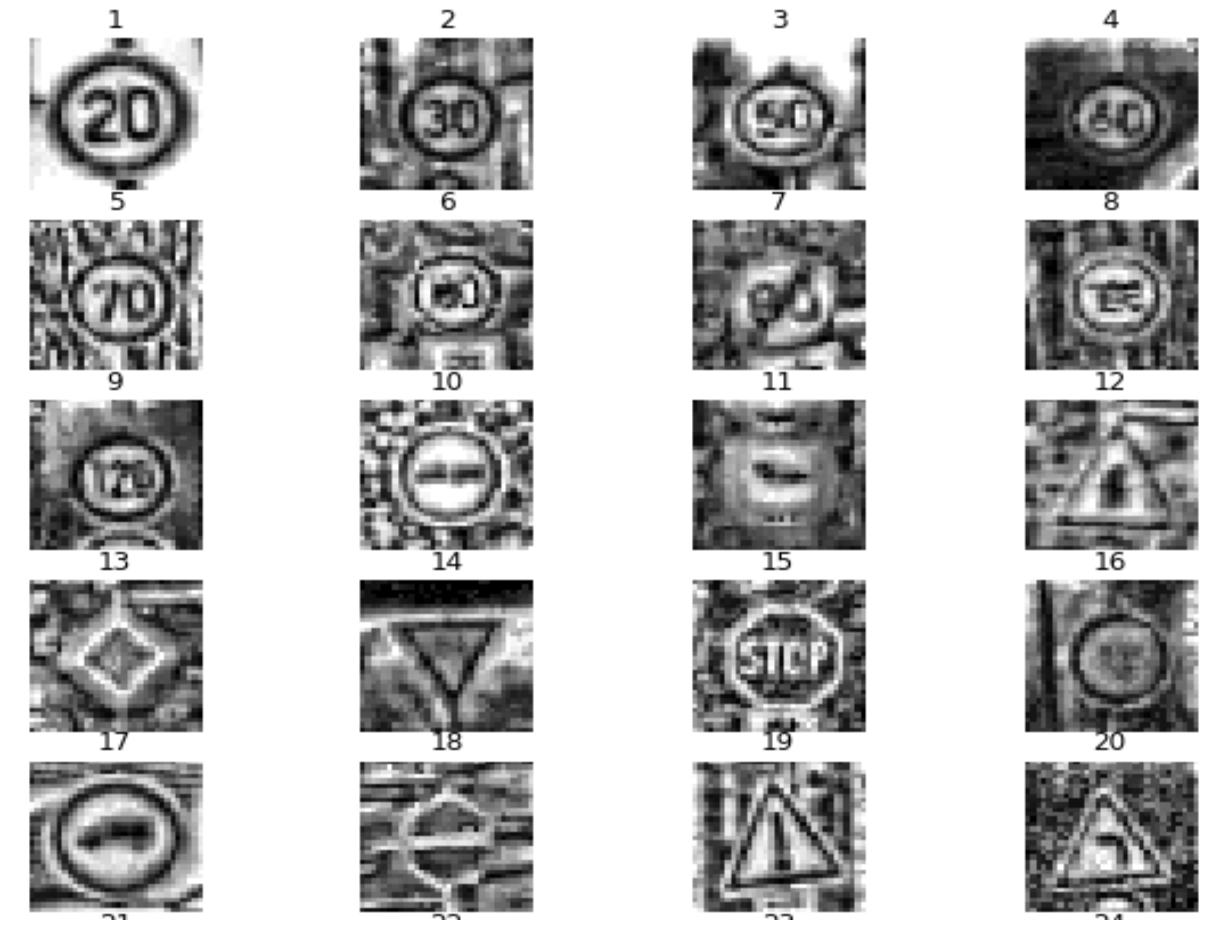
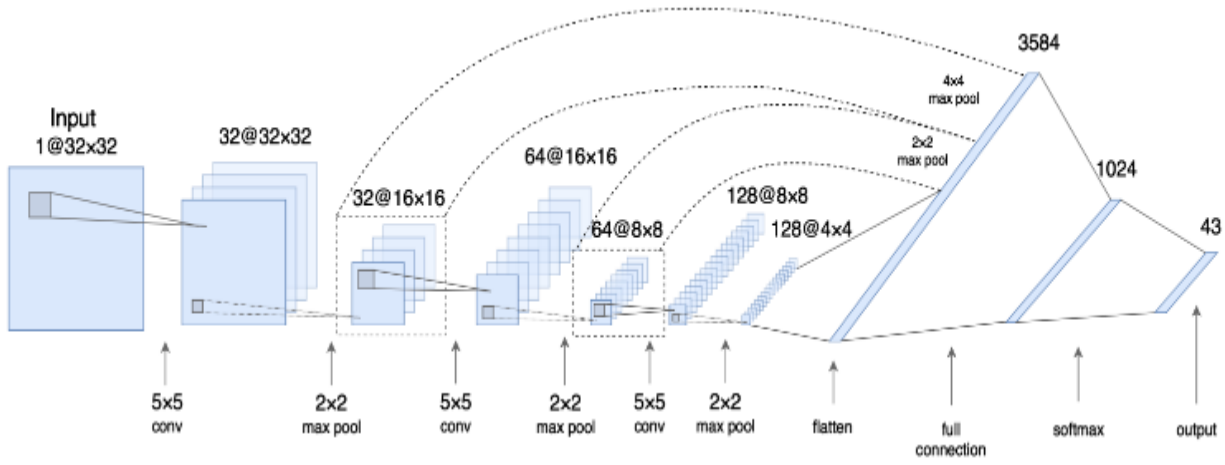


Figure: Histogram normalized examples

Design and Test a Model Architecture



My final model consisted of the following layers:

Layer	Description
Input	32x32x1 Normalized, hist. equalized image
Convolution 5x5	1x1 stride, same padding, outputs 32x32x32
RELU	
Max pooling	2x2 stride, outputs 16x16x32
Convolution 5x5	1X1 stride, outputs 16x16x64
RELU	
Max pooling	2x2 stride, outputs 8x8x64
Fully connected	3584 input, 1024 output
Softmax	1024input , 43 output

The model I used is inspired by "Traffic Sign Recognition with Multi-Scale Convolutional Networks" by Yann LeCun and Pierrre Sermanet and as implemented in http://navoshta.com/traffic-signs-classification/ . It has 4 layers: 3 convolutional layers and 1 fully connected layer as a classifier. It uses multi-scale features, where the convolutional layers' output is fed to both the subsequent layer and the final classifier after additional max-pooling. I used a different drop-out probability of 0.8 for each of the convolutional layer and the fully connected layer. Training was done using the adam optimizer till the validation set accuracy reached around 97% after 53 epochs with batch size 128 and learning rate of 0.001. The Test set accuracy achieved was 94.3%.

As mentioned above, I used the architecture from the paper "Traffic Sign Recognition with Multi-Scale Convolutional Networks" by Yann LeCun and Pierrre Sermanet. I chose this architecture as it was a fine tuned network specifically to work on traffic signs. It is an inspired from the lenet architectures with multiscale features. The implementation finished second place in the GTSRB challenge and hence I felt it would be the right fit. The validation set accuracy is 97.3% (exited due to early termination) and the test set accuracy is 94.3%. On the new user acquired test set, the accuracy was around 84%. This was due to the fact that I went out of my way to find images that would be hard for the network to classify.

Some German traffic signs that I obtained from google street view on the streets of Frankfurt are shown in block 18 of the ipynb notebook. I collected a few very good and clear signs and a few images with various amounts of clutter, distortion and backgrounds in the signs. These are also shown below.

Images 0,1,2,3,4,5,7 and 10 are well localized and have tight bounding boxes and hence should be easy for the trained network to classify. I was more interested in checking for non-ideal cases such as 4, 6 and 8 where there is a lot of background included, in 7 and 9 there are distortions and in 11, the sign is confusing as there is some extra marking apart from the ahead only arrow.

The predictions are shown below. The left columns are labeled with the predicted data and the right columns are the original ground truth data. As we can observe, most of the easy images were classified right. The images with large backgrounds were classified incorrectly and images with slight distortions were classified correctly.

Softmax probabilities for the test set is shown in block 49 of the ipynb notebook.