Traffic Sign classifier

Objective:

The goals is to classify German traffic signs and differentiate between 43 different signs

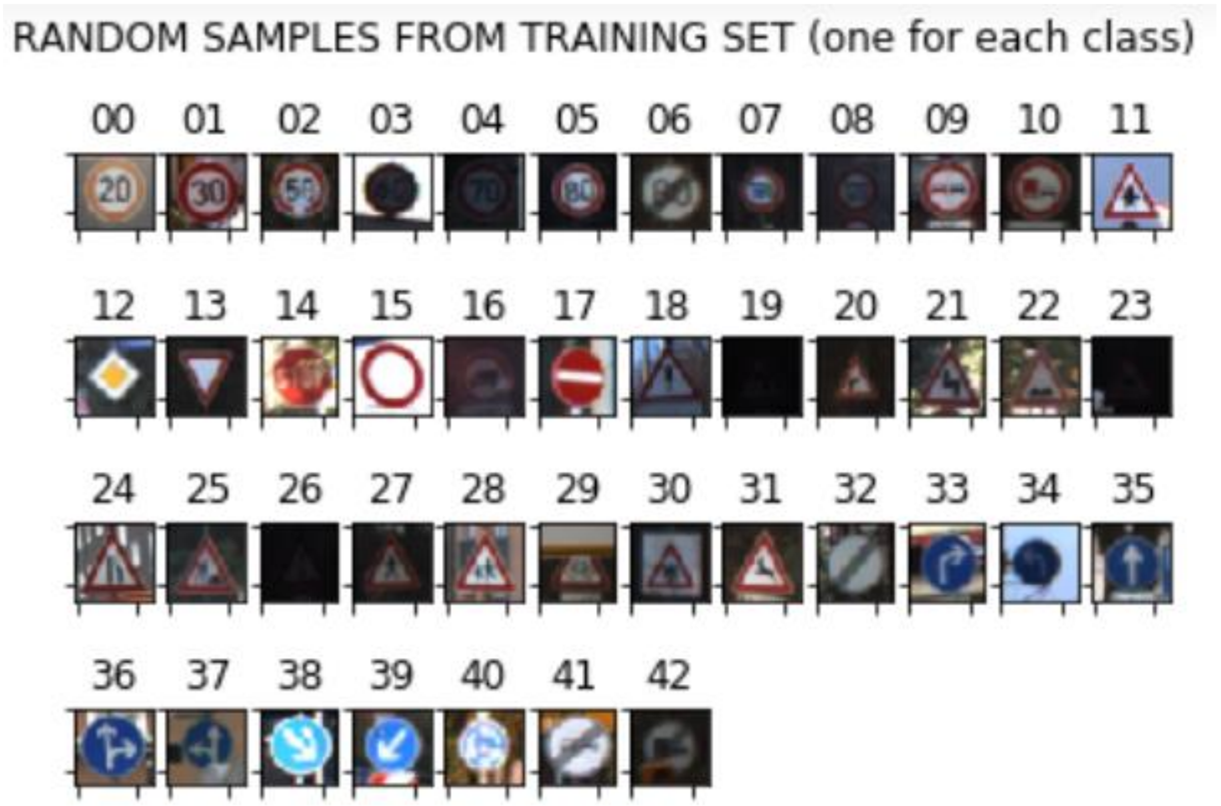
Data Set Summary & Exploration

Number of training examples = 34799

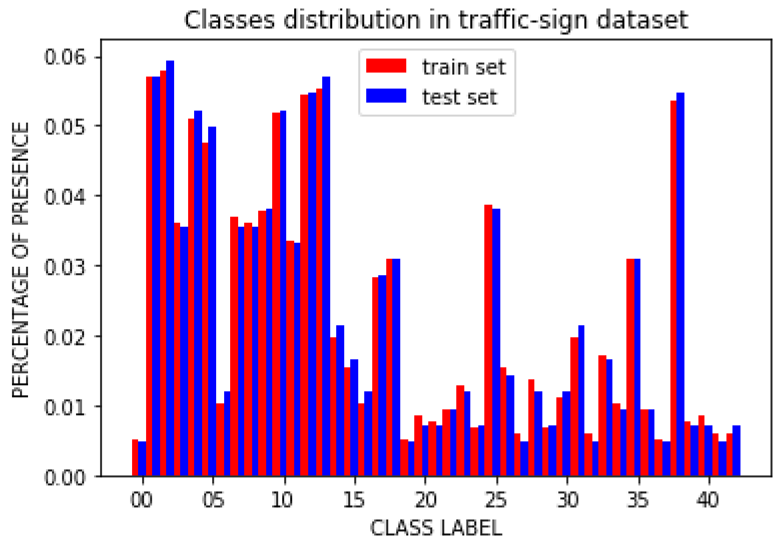
Number of testing examples = 12630

Image data shape = (32, 32, 3)

Number of classes = 43



Images labels Distribution:



Random images:

Design and Test a Model Architecture

Preprocessing:

I tried 2 techniques which are CLAHE (Contrast Limited Adaptive Histogram Equalization) and Keras generator images but I got poor results so I just normalized images and convert them to gray scale

Reasons: because I don't want the neural network to link between the sign and color because sometimes the same sign has different color

Model Architecture:

Implement the network architecture which consists of 6 layers with relu activation func except last one:
Layer 1: convolutional with input 32x32x1 and output 14x14x30 because I used valid padding then added max pooling
Layer 2: convolutional with input 14x14x30 and output 5x5x60 because I used valid padding then added max pooling
Layer 3: full connected with input 5x5x60 and output 2160
Layer 4: full connected with input 2160 and output 2000 and dropout = 0.7
Layer 5: full connected with input 1000 and output 100 and dropout = 0.7
Layer 6: full connected with input 100 and output 43 (no of classes)

Model Training:

The main hyper parameters that I was trying to tune were epochs,batch_size and learning rate

I tried to use more than 30 epochs but I found that the model is over fitting and with lower than 30 the model didn't reach the target accuracy

Also when I decrease the learning rate less than 0.001 the model was learning slow

For batch size I tried 64 but also the model was learning slow

So I used to Train, evaluate and test the model 30 epochs with batch_size = 128, learning rate = 0.001 using Adam optimizer:

Solution Approach:

I used Labnet5 model because it give good results and I tried to increase number of filters in the convolutional layers we decreasing the size of filters

Also I increased no of neurons in the Fully connected layer

I made these changes because I was classifying 43 classes which is harder that the example I had before in course which classify 10 classes

Tuning different no of epochs, control the size of fully connected layer, no of filters and use dropout to decrease over fitting

Also I printed the accuracy on training data and validation data each epoch to observe the model for not over fitting or under fitting

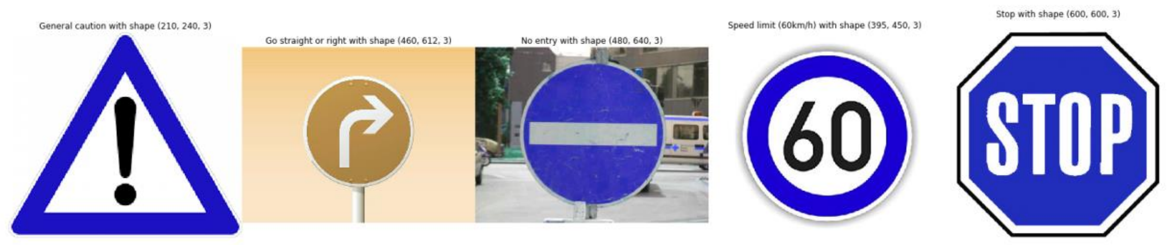
Important Notes:

I decrease the learning rate after 12 epochs for not making too much oscillation between under fitting and over fitting

I stop running epochs once I reach the accepted validation accuracy

Test a Model on New Images:

I searched on internet to get 5 random images with random sizes for 5 German traffic signs and here are the images I used and visualized them


General caution with shape (210, 240, 3)
Go straight or right with shape (460, 612, 3)
No entry with shape (480, 640, 3)
Speed limit (60km/h) with shape (395, 450, 3)
60
Stop with shape (600, 600, 3)
STOP

As we can see we have different size,resolution and color conditions but what is important is the size because we teaches the network to ignore color conditions by using gray images

I believe that the more the network depend on shape of sign (outlines) ignoring other factors the more it will be robous

I put all my test images in a folder then I read all images, before any processing I need to resize it to 32x32x3 for testing it

I feed the resized test images to logits func for predicting it

I calculate the accuracy on the 5 images by comparing what is predicted by my model with the labels that I set by myself

I printed out the labels under name real sign names and the predicted labels under name predicted sign names

The comparison is done on labels values between prediction and what I set

Test data accuracy was 0.917 while accuracy of my new images were 0.8 which means 4 images predicted correctly

I think this is a good result for initial network we may improve the test result quality if we feed the network with more than 34799 training examples as the web has a huge no of images

Also increasing the power of fully connected layers and no of filters will increase the accuracy but also we may fear from overfitting

Then I calculate the softmax of the 5 images

Potential shortcomings with your current pipeline

Increase the accuracy of the training