Traffic Sign Recognition

Build a Traffic Sign Recognition Project

The goals / steps of this project are the following:

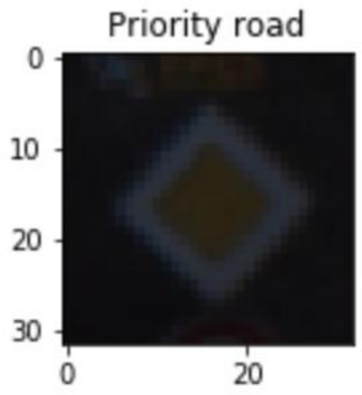
- Load the data set (see below for links to the project data set)
- Explore, summarize and visualize the data set
- Design, train and test a model architecture
- Use the model to make predictions on new images
- Analyze the softmax probabilities of the new images
- Summarize the results with a written report

Data Set Summary & Exploration

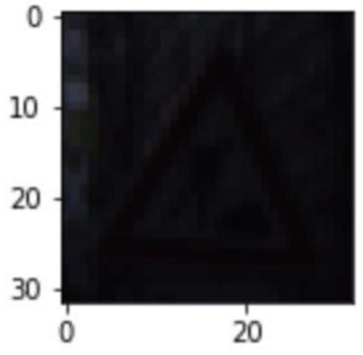
1. Dataset:
   To extract information about dataset, I've just used basic Python functions.

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

2. Visualization of the dataset:
   Before any operation, there are few examples of signs from dataset:

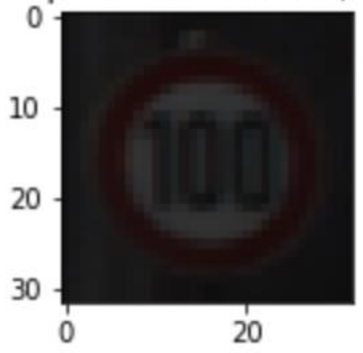


Road work

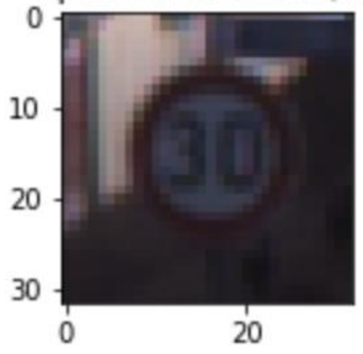


Stop

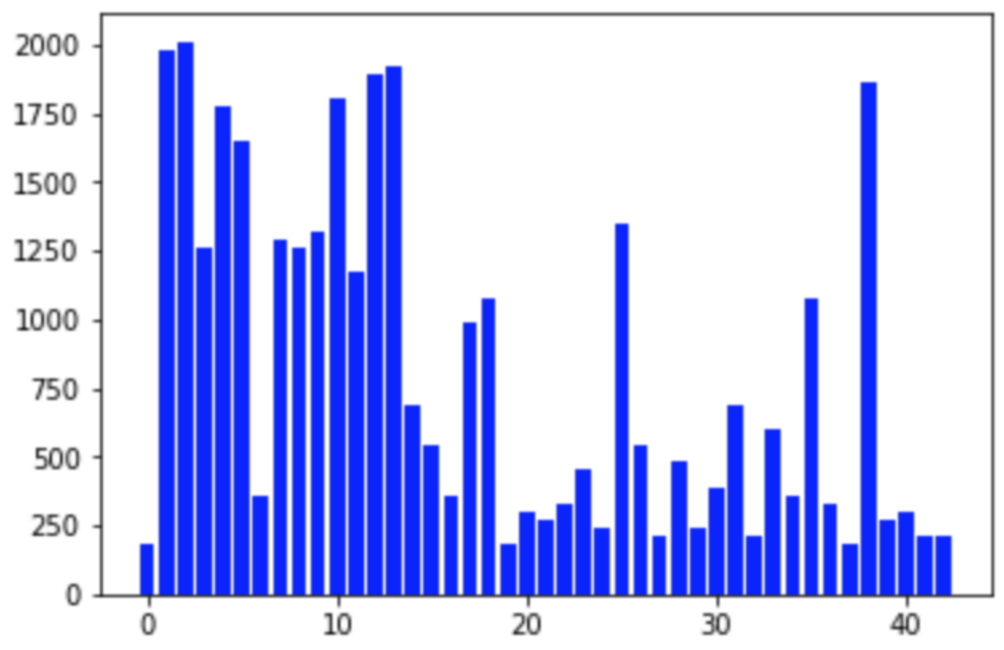
Speed limit (100km/h)



Speed limit (30km/h)



Data distribution for each class in data set is as follows:



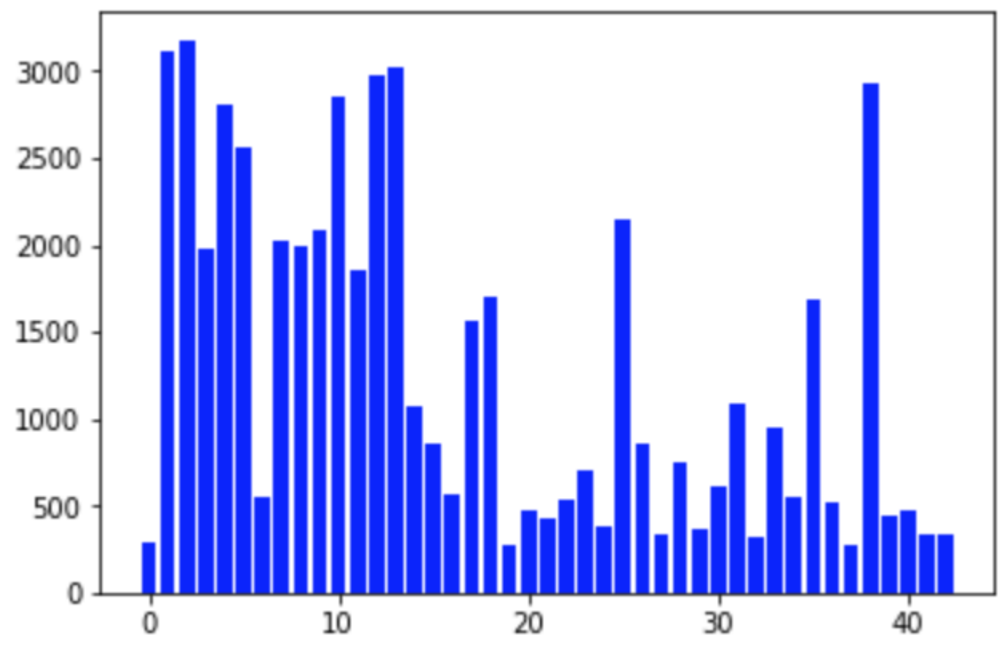
As we can see the dataset is very skewed. Some classes have lots of examples and some classes have very low examples. The model might not have enough examples for some classes making the learning process hard.

Design and Test a Model Architecture

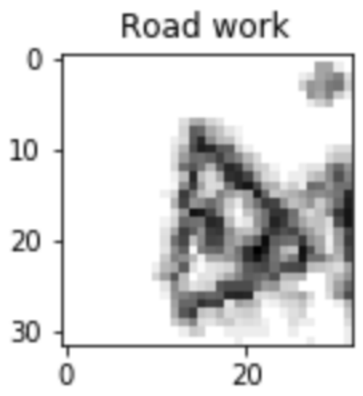
1. To mitigate the above issue I augmented some extra data for training using Keras's ImageDataGenerator() function. However I was not sure if augmenting extra data only for the classes with less data is actually a good idea. It might be the case that the skewed nature of the data points towards the occurrence probability of certain signs in Germany. Therefore I decided to randomly generate 20000 images to support learning process.

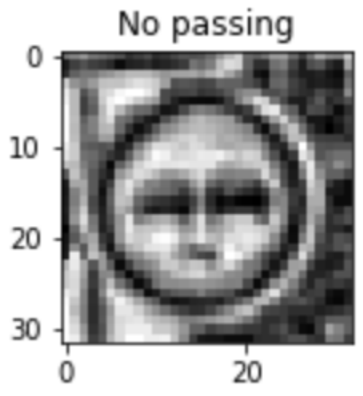
   Data generated after Augmentation:

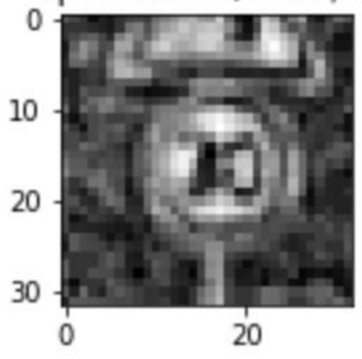
   ```
   Number of training examples = 54799
   Number of validation examples = 4410
   Number of testing examples = 12630
   Image data shape = (32, 32, 3)
   Number of classes = 43
   ```

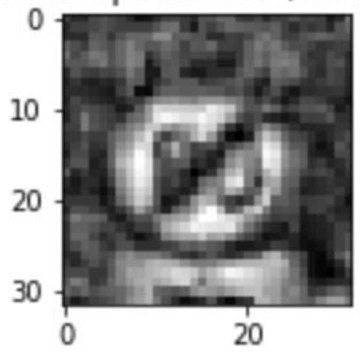


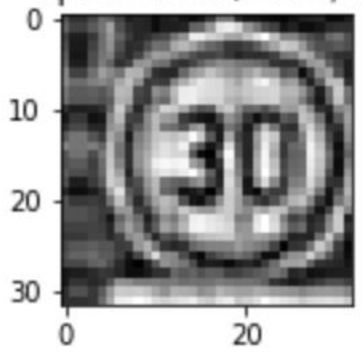
Then I pre-processed the images by converting them to grayscale and applying Contrast Limited Adaptive Histogram Equalization and Normalization. I applied grayscale as in traffic sign detection colors don’t help much and converting to single channel is more efficient. I applied histogram equalization and normalization to account for varied brightness and contrast in images. Below you can see few processed examples.






Speed limit (30km/h)


End of speed limit (80km/h)


Speed limit (30km/h)

2. The Final model architecture is as follows:

Layer	Description
Input	32x32x1 RGB image
Convolution 5x5	1x1 stride, valid padding, outputs 28x28x6
RELU	
Max pooling	2x2 kernel size, 2x2 stride, input = 28x28x6, outputs 14x14x6
Convolution 5x5	1x1 stride, valid padding, input = 14x14x16, outputs 10x10x16
RELU	
Max pooling	2x2 kernel size, 2x2 stride, input = 10x10x16, outputs 5x5x16
Flatten	input = 5x5x16, output = 400
Fully connected	Input = 400. Output = 120
RELU	
Dropout	keep probability = 0.75
Fully connected	Input = 120. Output = 84
RELU	
Dropout	keep probability = 0.75
Fully connected	Input = 84. Output = 43
Softmax	

3. Model train parameters.

Implemented LeNet architecture with Adam optimizer with below hyperparameters:

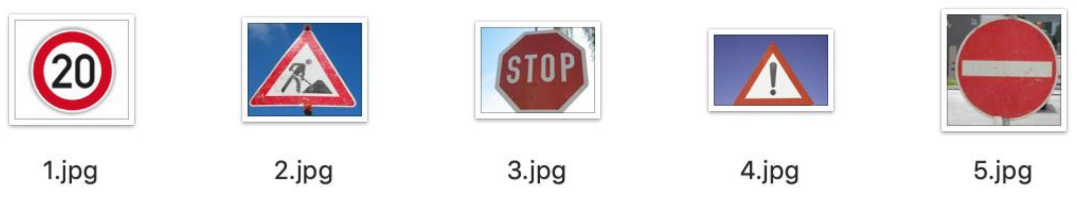
- Batch size: 32
- Epochs: 70
- Learning rate: 0.001
- Mu: 0
- Sigma: 0.1
- Dropout keep probability: 0.75

4.Discussion

Firstly, I used the classical LeNet architecture with the provided dataset and got an accuracy of around 91% on validation set. However it quickly became clear that more data was needed and therefore I applied data augmentation to create 20000 extra data examples. Along with this I also experimented with adding dropouts to fully connected layers and different values for hyperparameters. Finally I received an accuracy of 96% on validation set and 93.7% on test set . I plan to conduct more experiments by modifying layers of the architecture. Also it seems that the model will improve if I can augment or find more data for classes containing less images.

Test model on New Images

I tested my pipeline with few images for german traffic signs from the web.



I applied same pre-processing to these images and fed them to the pipeline for prediction. The predictions are as follows:

Image	Prediction	Probabilities
No entry	No entry	1
Stop	Ahead only	0.63
General caution	General caution	1
Speed limit (20km/h)	Speed limit (20km/h)	0.99
Road work	Road work	1

The fact that stop sign was at an angle in the picture and a bit zoomed in might have prevented it from being detected correctly. Perhaps I need to experiment with different data augmentations to account for such images.

The pipeline was able to predict 4 out of 5 signs correctly. This gives model an accuracy of 80% which is less than the accuracy of the test set. It's clear that the road signs found in real time using a car mounted camera under difficult lighting conditions will be a tough task for this model. More sophisticated architectures and techniques are therefore required to make this model useful for real world use.