Project 3: Traffic Sign Classifier

Dataset Summary:

I read the dataset using the pickle library in Python. The original dataset features:

1. Number of training samples is 34799
2. Number of testing samples is 12630
3. Number of validating examples is 4410
4. Image data shape = (32, 32, 3)
5. Number of classes = 43

Visualisation of the dataset:

The number of examples of each class has been represented in the bar chart below. Many classes have very less data as compared to other classes. Hence, I have used the function 'Generatedata' to make sure that all classes have at least 800 examples. Generate data calls two functions 'warp' and 'scale' which randomly apply transformations on images. A few examples have been shown below.

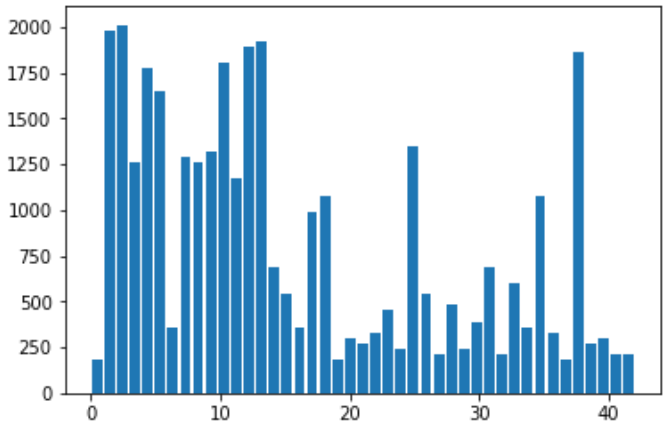


Figure 1 Bar Chart

The original dataset comprises of images as shown below. The label of the image has been indicated above them.

Figure 2 Examples

Pre-processing of dataset

Pre-processing the original is a very important technique in Machine Learning. In my model I decided to convert the image into grayscale. On observing the results, I realised that some images were simply too dark. I found it difficult to myself classify these traffic signs. To further process the images, I used an OpenCV function CLAHE (Contrast Limiting Adaptive Histogram Equalisation). It greatly improved the image contrast. I would like to draw your attention to the traffic signs in green box.

Figure 5 Original

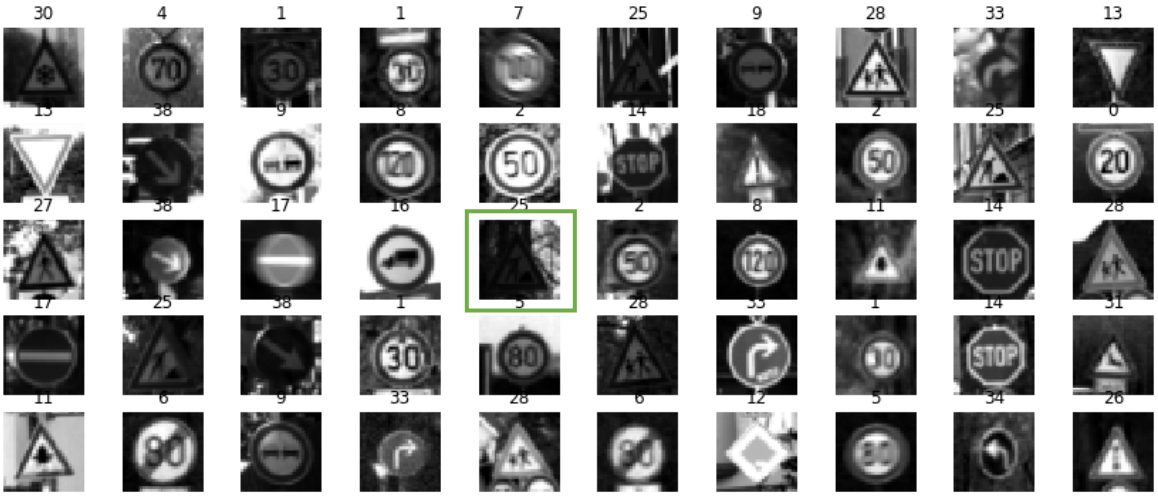


Figure 4 Grayscale

Figure 3 CLAHE

The example in green box is very faint in the original dataset. After converting it into grayscale, it is barely recognizable. However once ve apply CLAHE, the sign is quite clearly visible.

The data is then standardised so that all the values lie between 0 and 1.

The image below shows some examples of images generated by the 'Generatedata' function:

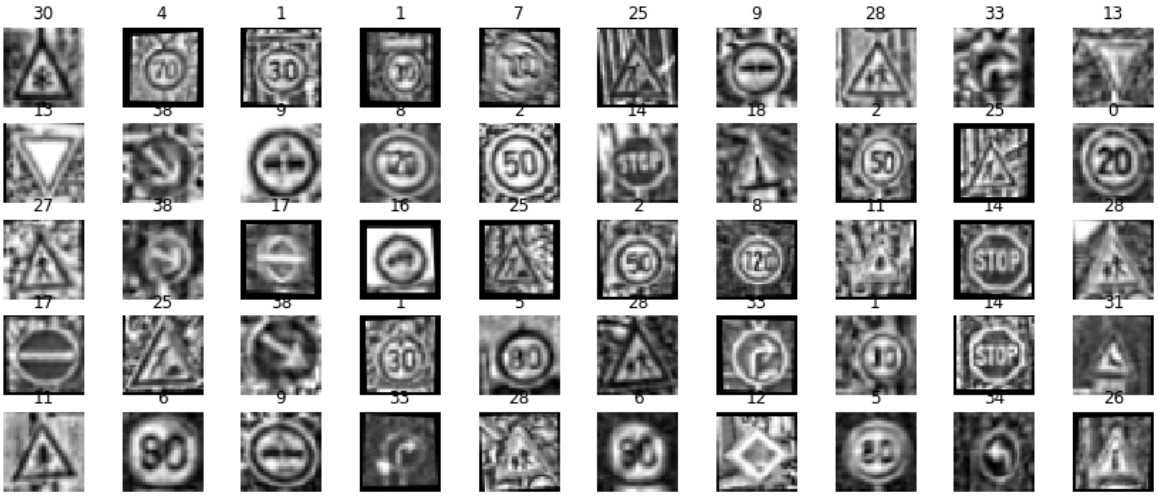


Figure 6 Generated data

Model Architecture

The architecture of the model is:

- Layer 1:
  - Convolution with input 32x32x1 and output 28x28x6
  - Activation using RELU
  - Max pool with output 14x14x6
- Layer 2:
  - Convolution with input 14x14x6 and output 10x10x16
  - Activation using RELU
  - Max pool with output 5x5x16
- Layer 2b:
  - Convolution with input 5x5x16 and output 400
  - Activation using RELU
- Dropout:
  - Probability of .75 produced best results
  - Outputs of layer 2b and layer 2 were fed into this step
- Layer 3:
  - Fully connected with input 800 and output 400
  - Activation using RELU
- Layer 4:
  - Fully connected with input 400 and output 200
  - Activation using RELU
- Layer 5:
  - Fully connected with input 200 and output 43
  - Activation using RELU

Model Training

The following parameters were used while training the model:

1. Epochs = 30
2. Batch size = 128
3. Learning rate = 0.001
4. Optimizer: Adam Optimizer
5. Dropout keep probability = 0.75

Solution Approach

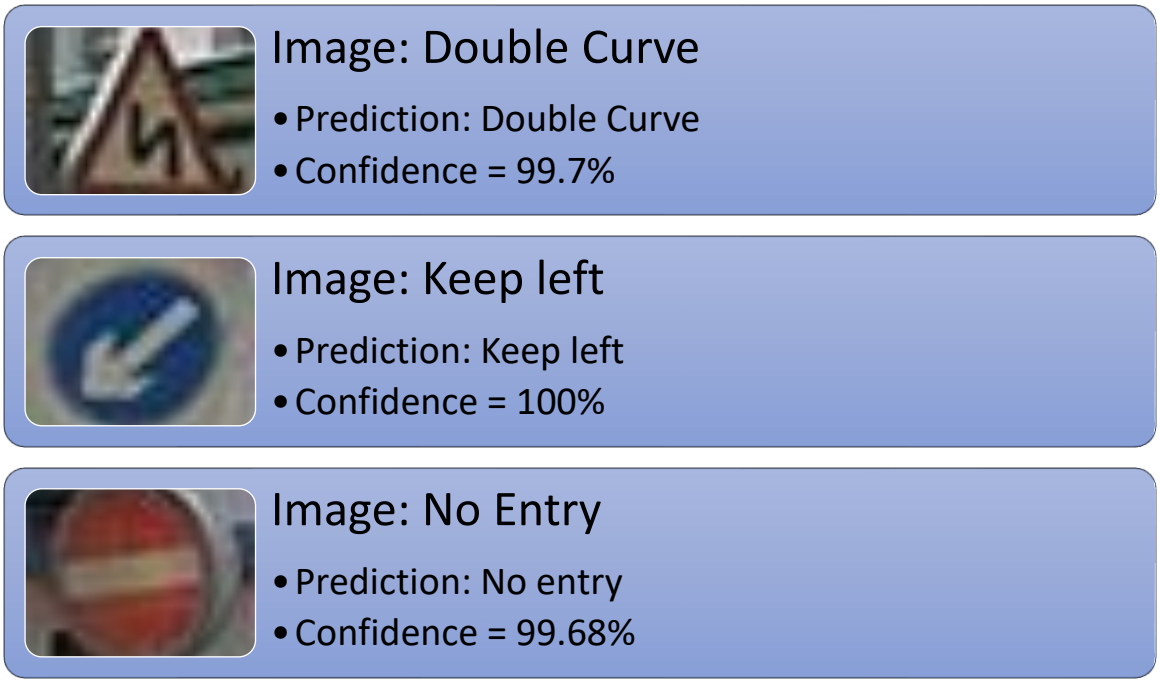
The result is:

1. Validation accuracy = 95.2%
2. Test accuracy = 93.3%

I had earlier built a model where I used colour images as input for the model. However, using grayscale images provided better accuracy.

The model I was using was the same as the one used by us in Convolution Neural Network. However, using the model similar to Sermanet/ LeCunn's model provided better results

Performance on new Images



A total of 10 images were provided. All of them were predicted correctly with very high confidence.