PROJECT REPORT

UDACITY

SELF-DRIVING CAR NANODEGREE

Project 3: Traffic Signs Classifier

AUTHOR:
Thomas Teh

Date: January 26, 2019

1 Introduction

This project is part of the Self-Driving Car Nanodegree by Udacity and its objective is to train a traffic signs classifier. The data set used to train the model is the German Traffic Sign Dataset and the summary of the data is given below:

Type	Amount	Image Size	Color
Training	34799	32×32	RGB
Validation	4410	32×32	RGB
Test	12630	32×32	RGB

The dataset contains 43 different classes and samples of the training images are shown in Figure 1. Also, we visualize the distribution of the different classes and we can observe that there is a significant class imbalance as in Figure 2.

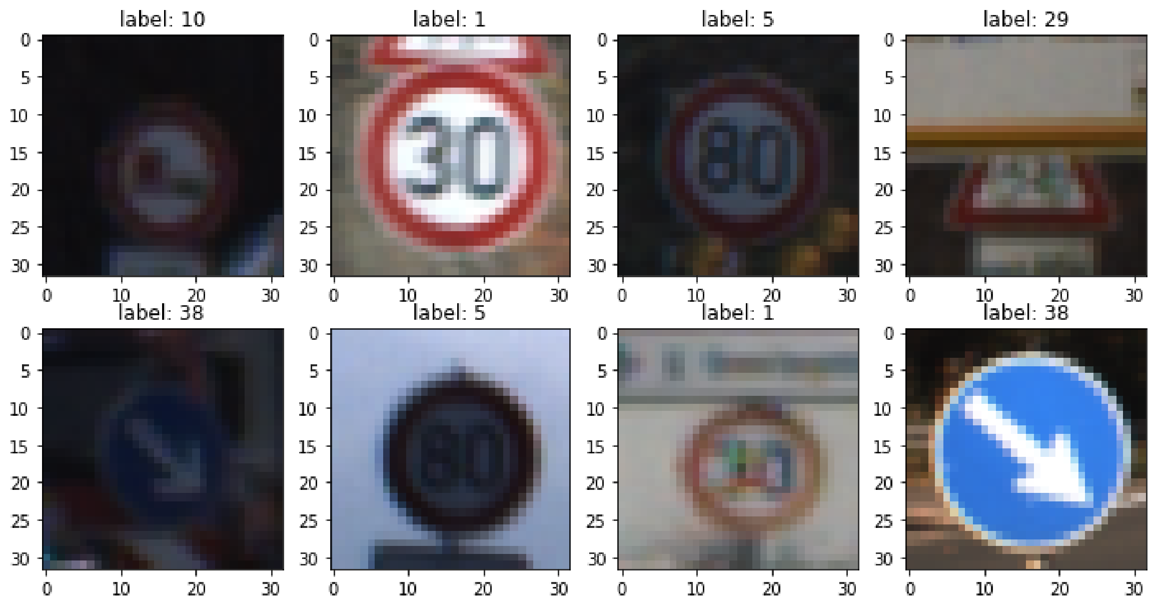

Figure 1: Sample training images.

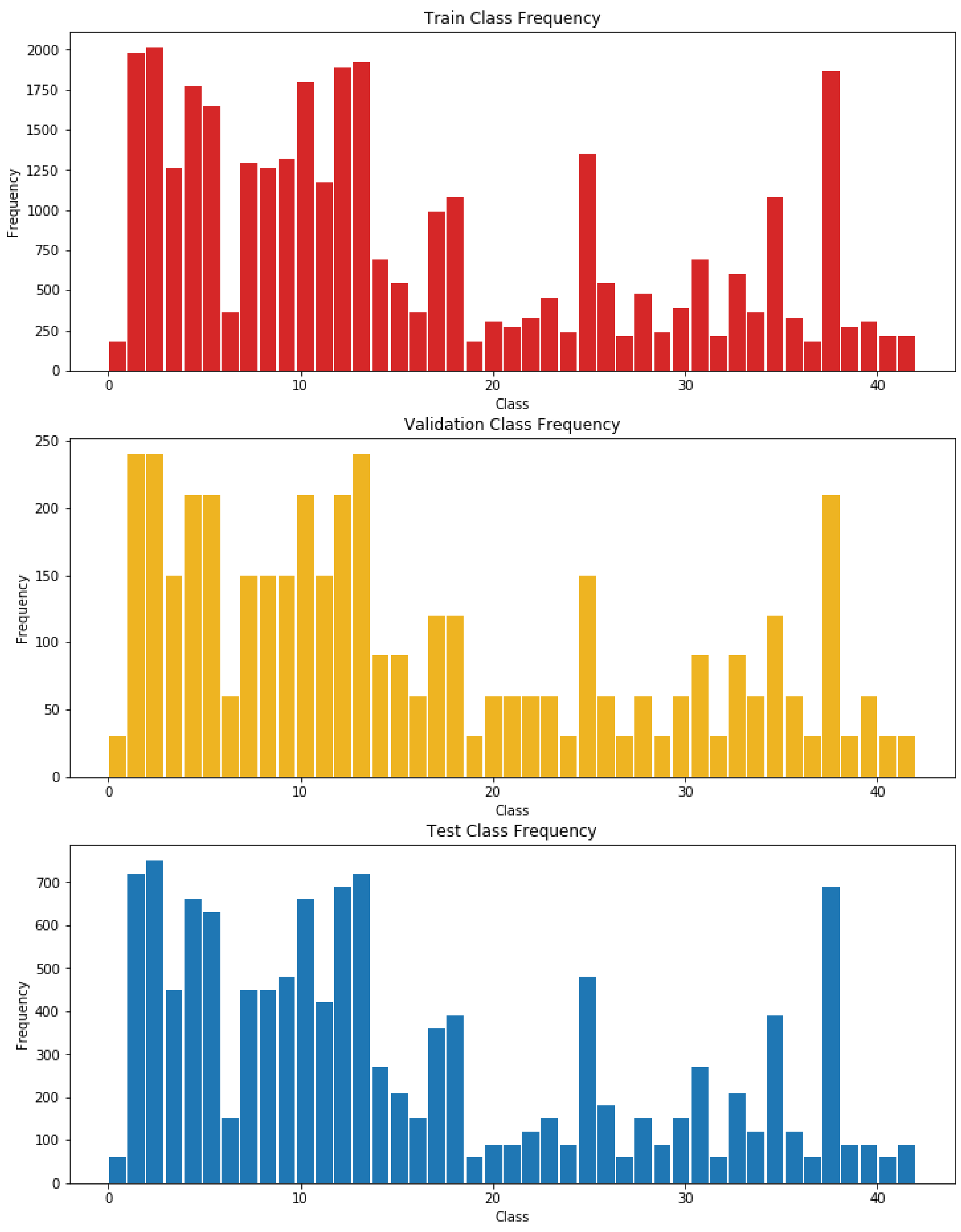

Figure 2: **Top:** Training Set **Middle:** Validation **Bottom:** Test

2 Image Preprocessing

Preprocessing is done in the following steps:

1. **Data Augmentation:** We augment the data by random performing the below transformations on the original training data set:

   - Rotation: 25 degrees
   - Width shift: 0.20
   - Height shift: 0.20
   - Zoom range: (0.80, 1.20)

   With the augmentation, the amount of the training data available is tripled to 104,397.

2. **Convert to Grayscale:** Since the colors of the signs do not make a big difference in terms of distinguishing the them, we convert the RGB images into grayscale. The advantage of this is it reduces computation, hence allow us to use a much deeper model without having to do distributed training on the data set.

3. **Histogram Normalization:** I applied the Contrast Limite Adaptive Histogram Equalization (CLAHE) to make the edges of the signs more obvious.

A sample of an image after preprocessing is shown in Figure 3

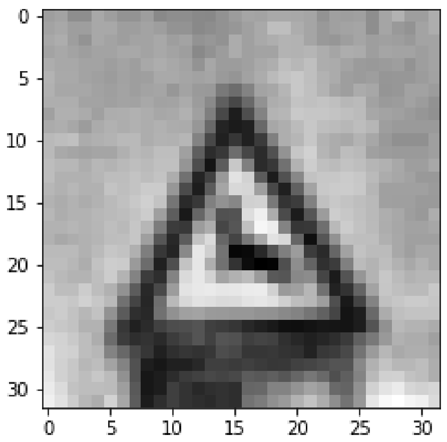

Figure 3: Image after preprocessing

3 Architecture and Hyperparameters

The architecture that I used is shown below:

Input
Convolution Layer Kernel size: 5×5, N Kernels: 256
ReLU Activation Layer
Max Pooling Layer Kernel size: 2×2
Convolution Layer Kernel size: 5×5, N Kernels: 512
ReLU Activation Layer
Max Pooling Layer
Kernel size: 2×2
Convolution Layer Kernel size: 5×5, N Kernels: 2048
ReLU Activation Layer
Max Pooling Layer Kernel size: 2×2
Dense Layer: 1024 units
ReLU Activation Layer
Drop Out Layer: dropout prob = 0.60
Dense Layer: 512 units
ReLU Activation Layer
Drop Out Layer: dropout prob = 0.60
Dense Layer: 43 units

The model is trained with ADAM optimization algorithm with the following hyperparameters:

Hyperparameters	Values
Epoch	20
Batch Size	128
Learning Rate	0.0010

Other comments:

- Batch normalization did not help in the training process significantly and it results in a much slower training process. Hence I did not implement batch normalization in the final model.
- ELU activation works but they did not perform as well as ReLUs. Therefore, I decided to stick to ReLU activation unit.
- Drop out layers are used to prevent overfitting since we have a large number of neurons in the dense layers.

4 Performance

The performance of the model after 20 epochs of training is shown in the table below:

Data Type	Accuracy
Training	99.96%
Validation	98.91%
Test	98.20%

In addition to the test data, I sourced 6 images from the web and predictions from the model. The model achieves a 83.33% accuracy. The model misclassified 1 image out of 6 images, and the reason is likely that this image contains a traffic sign that is vandalized (see Figure 5).

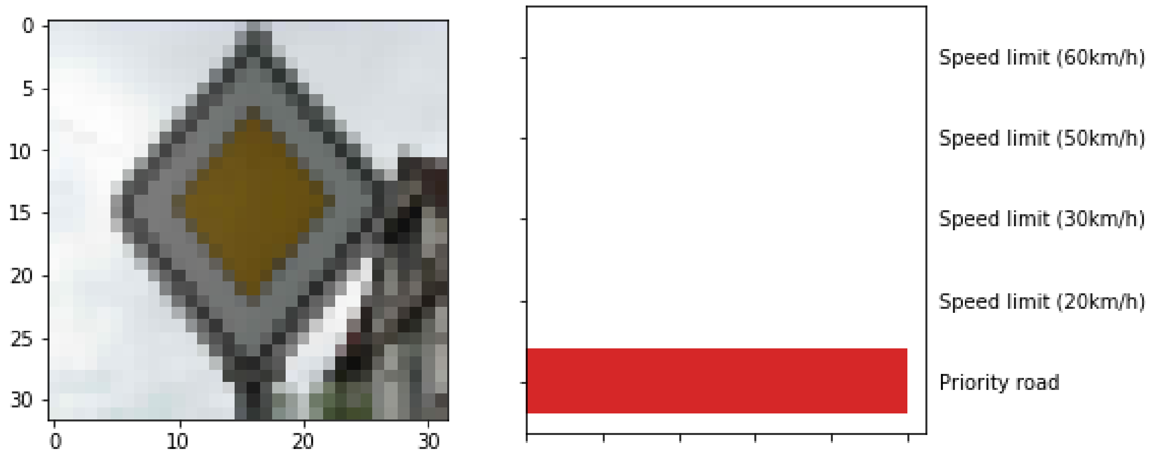

Figure 4: Image that was classified correctly.

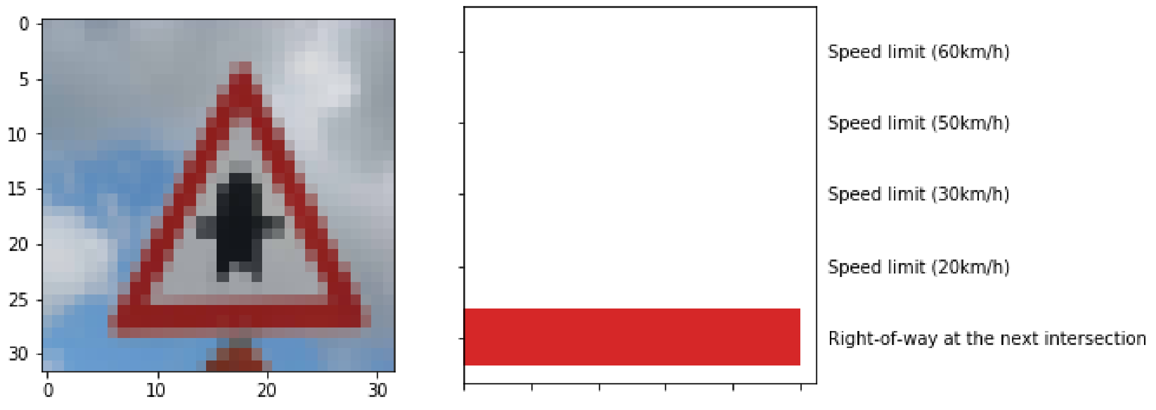

Figure 5: Image that was classified correctly.

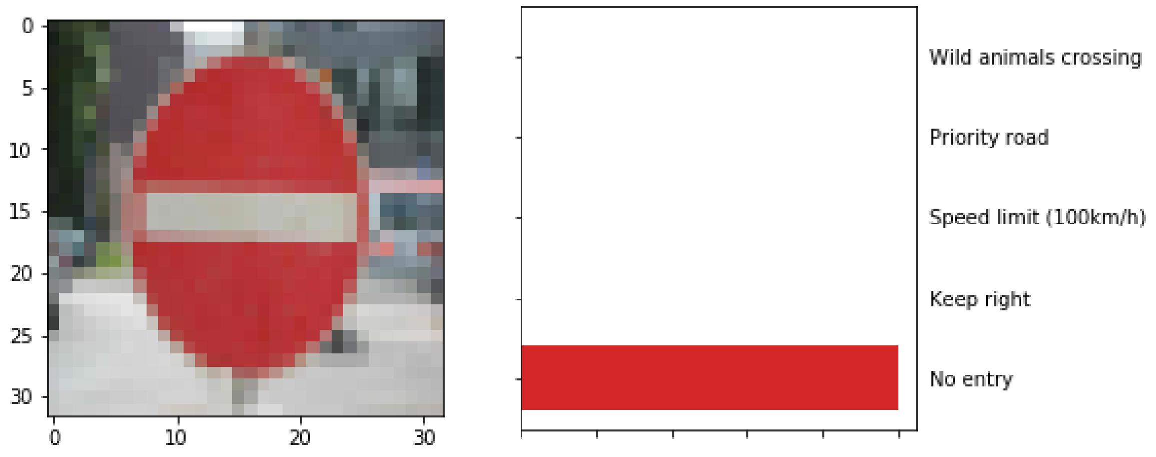

Figure 6: Image that was classified correctly.

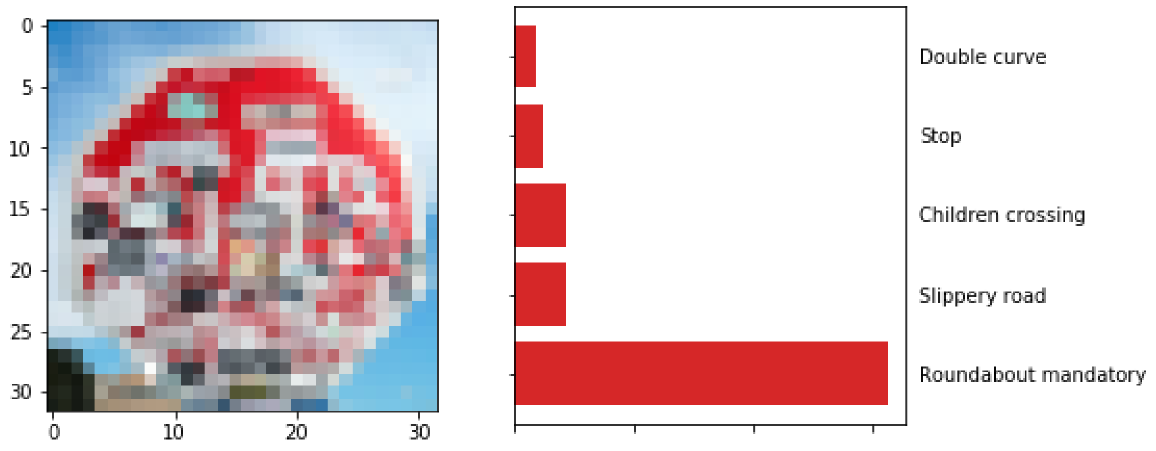

Figure 7: Image that was misclassified.

5 Ideas for Improvements

The following are some ideas for improving the project:

- Can potentially do transfer learning. Instead of training a model from scratch, we can used a pretrained model such as ResNet or VGGNet. We can freeze the layers at the start of the model and only train the classifier. Given these models are trained on a larger data set, I would expect that the models developed using transfer learning to perform well.
- Adjust the training process to account for the class imbalance. This will improve the performance of the model.
- Expand the training data set by collecting the more data set. Most traffic signs in the German Traffic Signs data are similar or the same as those in other countries. We can potentially augment the data with real images of traffic signs from other countries.