Capstone Project 2 - Traffic Sign Image Recognition using Convolutional Neural Network

Milestone Report 1

Project Background

Reading traffic signs is an important component of autonomous driving. Autonomous cars need to be able to see the road signs just for the same reasons humans need them. They allow the driver to know the speed they should be traveling, where to stop, and what to be cautious of on the road ahead. Automotive companies could use traffic sign recognition as an aid to human-driven cars as well. Sometimes a human driver can miss a sign alerting them to an upcoming curve or a speed limit change, and the car could remind the driver if it doesn't sense proper responses to signage.

Data

The data to be used for this project comes from the Institute for Neuroinformatik in Germany. This data can be accessed at the INI website:
http://benchmark.ini.rub.de/?section=gtsrb&subsection=news.

The data set contains 35,000 training images and 12,000 test images of 43 classes of German traffic signs. The images are 32 x 32 pixels and have 3 color channels.

Data Exploration

First I will plot some sample images and their classes to get an idea of what the data looks like.

The images are of low resolution at only 32 x 32 pixels, and the number of images is not very large.

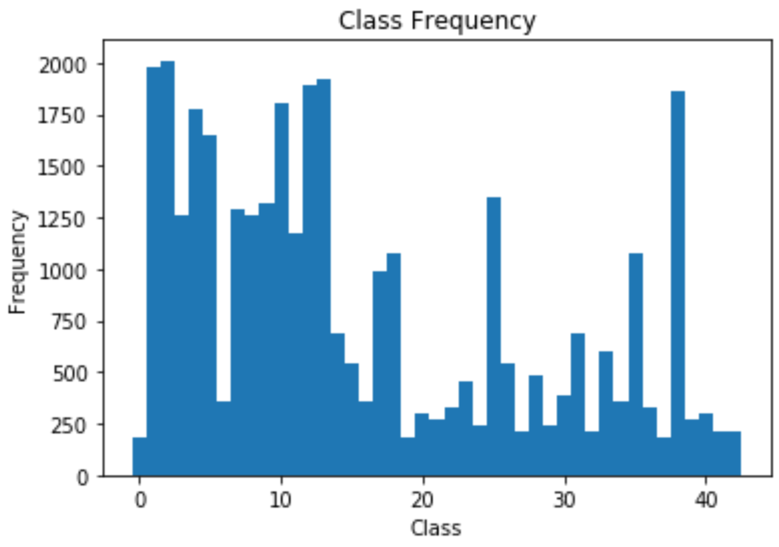
The 43 classes of traffic signs include speed limit signs, as well as various warning signs. The speed limit signs look identical except for the speed number in the center. Otherwise they are a white sign with a red border. Due to this similarity, the neural network may find these signs particularly difficult to distinguish.

There are also warning signs that look similar, with a red border around triangular white sign. Only the image in the center of the sign varies, such as an exclamation point of a wild animal.

The sample images appear to be fairly well centered in the image, with only small amounts of rotation seen. Also, the images appear to all be taken from a nearly head on angle.

Class Imbalance

The classes are heavily unbalanced, with some classes having more than 2,000 training images, while others only have a few hundred.



While training, the network will pull a random sample batches to pass, and it may take a long time before it sees enough of the classes without many images. To balance the training data, I created a function that determines how many samples in each class to upsample, and upsamples them while also applying a random rotation within a specified number of degrees. Adding this rotation should allow the model to generalize on new data a little better.

Convert to Grayscale

For the first training passes, I will convert the images to grayscale. This will help speed up the training, and may actually help the model since much of the color variation is in the background. Also, grayscale images will be better suited for further preprocessing, such as histogram equalization. Future training phases could included training on the full color images, which could also include further preprocessing the images to remove the background.

Next Steps

The next steps will be to start building the model architecture for the neural network, and begin training the network and tuning the hyperparameters.

The baseline model architecture will be two convolutional layers, followed by a flattening layer and three fully connected layers, the final of which will be size 43 to represent each of the classes the model is trying to predict.