Recognizing Traffic Signs

Why would we want machines to be able to read street signs? Just a few years ago the answer would seem like science fictions. Self-driving cars are becoming more and more a normal feature in cars. However self-driving cars aren’t the only use for having a computer recognize a sign. Having traffic cameras looking for signs that cannot be recognized could be used to know when to perform maintenance, cleaning, or replacement of traffic signs. So how do we make so a computer can read traffic signs?

The data set we have for this has a total of 51,839 images with 43 traffic signs of different lighting, angle, and quality. The data is separated into train, validate, and test sets 34,799, 4410, 12630 records respectively. The feature array in the data is thirty-two by thirty-two by three. This can be read as three sets of thirty-two by thirty-two arrays represented by numbers that are the color values of Red, Green, and Blue. When read together it creates the color image of the traffic signs. However, a computer has a very hard time seeing patterns when there are so many different numbers and values. To fix this we need to make all the colors all different shades of the same color. To do this I first divided all the values in the array by two-hundred fifty-five to get the data ready to be multiplied by values that would bring all the colors into grey scale. However, there was an issue the color changed to mono red instead of grey. Now this isn’t a huge issue as the data is all the same color which was the intended goal, but just not the color I was going for. Also, I tried many different ways to get the wanted color of grey, and was not able to achieve that. Now that the data uniform red it was ready to be put through a model.

The model I chose to use a Tensorflow Classification model. I chose the Tensorflow model because in an early class we saw a demonstration on a neural network classifying clothing and I wanted to use Tensorflow more as I have only been able to use it once before. The first thing I needed to check as the input data and the output data were the same shape. I this case the data was already in the desired shape. For the actual model I used a Sequential that flattens the arrays into a one-dimensional array with 300 hidden layers, 100 hidden layers and result of 43. The optimizer I chose to use for the training was adam and the measurement was accuracy. On the first run through of the model with an epoch of 30 the highest accuracy was 85% on the validation data. Not bad for a first try. From here I wanted to tune the optimizer. I created a GridSearchCV to ran all the optimizers that Tensorflow has to see which one was performing the best. The GridsearchCV returned nadam was the best optimizer being just a little better then adam. With that new information I created a new model using the better optimizer and the accuracy improved to 87%. With that model I then evaluated it on the test data set. The final result was an 83% accuracy. After all of this I wanted to see if K-Nearest Neighbors would be able to classify the signs. Unfortunately, it was only a 30% accuracy so I did not look into it any further. From here I wanted to see a prediction of the traffic signs from the test set to see what signs were being predicted correctly. Out of the small sampled pulled the signs that seem to predicted incorrectly are speed limits. This makes sense as all those traffic signs are the same with different numbers. This does bring up a concern, is an accuracy of 83% good enough.

The question is 83% accuracy good enough, and the answer is it depends. If we are using this to determine to send a crew out to replace a sign due to it not being readable anymore. Yes! 83% accuracy is great. Most upper management I have delt with live and die by the 80/20 rule. In the fraud department I work in we say the 20% of fraud not caught is what reminds the executives why they need a fraud department and might be more important then the 80% of fraud we stop. The issue is if you are using this in a self-driving car. 83% accuracy and that in the sample the traffic signs that are being misclassified are speed limits is a little worrisome. Now you can justify that the driver in a self-driving car would be able to correct for these misclassifications, but as more cars become self-driving more drives are going to just let the car drive. I would the test results in the 90s for accuracy and the for the traffic signs that are being misclassified to be signs that aren’t seen as often as speed limits. If this was a project that I was overseeing for a self-driving car I would say that 83% accuracy is not good enough to be used in the real world, but is a really good starting point and more development would be needed.