# Analysis Overview

The primary purpose of this project was to create a convolution neural network that can identify street signs in Germany. The goal was to be able to create a convolutional model that we could feel confident enough to deploy into a self driving car. We’ve tested and tweaked the model as well as enhanced the data in order to ensure that the model is as intelligent and powerful as it can be.

A few key points of this study are as follows:

* + - model was able to predict signs in the test data 99.5% of the time.
    - I tested over one hundred and twenty thousand different images. Of those images, it was correct 98.5% of the time. Of the test folder’s twelve thousand six hundred and 30 images, I was able to determine the sign 99.5% of the time.
    - loss is smooth, consistent, and improves continuously.

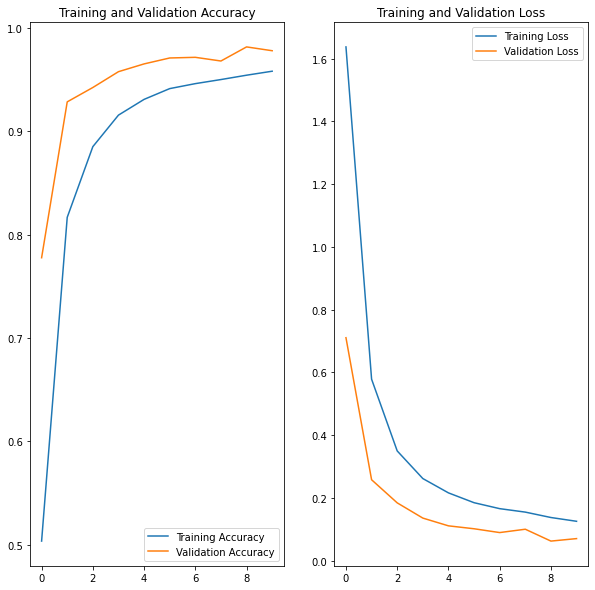
# Model Design

model was designed around some examples from the TensorFlow documentation and then modified to better suit data and the size of the dataset. I decided to increase the number of perceptrons per layer in model because I was working with a much bigger dataset than the examples in TensorFlow. I also had 43 potential categories to choose from so final output layer of the model needed to account for each of these.

Once model was built and I began testing, I noticed that there was some room for improvement in model’s design. Normalization of the color was the first thing I did. Standard RGB is measured from 0 to 255. Typically, these neural networks work best with “normalized” numbers between 0 and 1. I added a layer into the model that ensures the colors in each image processed ranged from 0 to 1. This increased model’s accuracy by an average of about .2%

The second thing I applied was a randomization layer in model. I essentially took every image and created a copy of it that was flipped, zoomed in on a random point or rotated in order to increase the number of training images model had the chance to process. This seemed to increase model accuracy the most with about a 1% increase in accuracy on average from before.

The final modification I made to model was to add a dropout layer into the model. This layer was inserted in order to help prevent model from overfitting to the data that it was training on. This ensured that I would get a very similar accuracy in test data as I did when fitting the model. You can see from the graphic below that accuracy and loss curves are both fairly smooth. The blue (training) and orange (validation) lines follow essentially the same arc. The orange line was originally very uneven with several peaks and valleys. There was not as consistent of an improvement over time until I implemented the dropout layer.



# Predictions

Initial Predictions were pretty good but accuracy when using test data was pretty random and jumpy. It seemed that overfitting was becoming a bit of a problem. I was able to smooth that out through data augmentation.

Accuracy started in the range of 96 to 97 percent. In order to improve upon this, I added the design changes mentioned above. With these changes, I managed to get accuracy for the validation data up to about 98 percent. When testing actual test data, model predicted 99.5% of all signs correctly.

Conclusions and Caveats

Overall, I was very impressed with the ability of model to accurately predict the type of street sign while “out in the wild”. A 99.5% accuracy rate is quite good and I feel that the model I finished with was significantly improved from the work I started with at the beginning of this module.

To further develop this project, it would be interesting to begin throwing in street signs from the United States into testing data. I currently do not have an “other” category and so model would try to characterize these signs into one of the available categories. It would be interesting to see if there are similarities between these signs and see how well model can recognize that a particular sign is not truly a match for one of the provided categories.