Bipin Sharma

Traffic Sign Recognition - Final Report

Problem Statement

One of the hottest topics of the last few years has been electric, self-driving smart vehicles. What's the one most important feature of self-driving cars? They are able to recognize the traffic signs accurately. And by accurate here I mean they need to recognize traffic signs 100% of the time - there is no room for errors, especially considering human life is in the balance. Creating a model able to recognize all the traffic signs will help deliver safer cars.

The traffic sign data needs to be fed back to the user of the car in real time in case of a non-self-driving car. This ensures that the user is alert. For example, some cars come with the speed recognition feature in the car. The car recognizes the speed limit on the road and feeds it back on the dashboard for the user to be aware (in case they missed it). This feature becomes even more important when it comes to self-driving cars, in which case the car is in control. This will help the car stay in lane, stay within the speed limits and take appropriate actions corresponding to the sign it encounters.

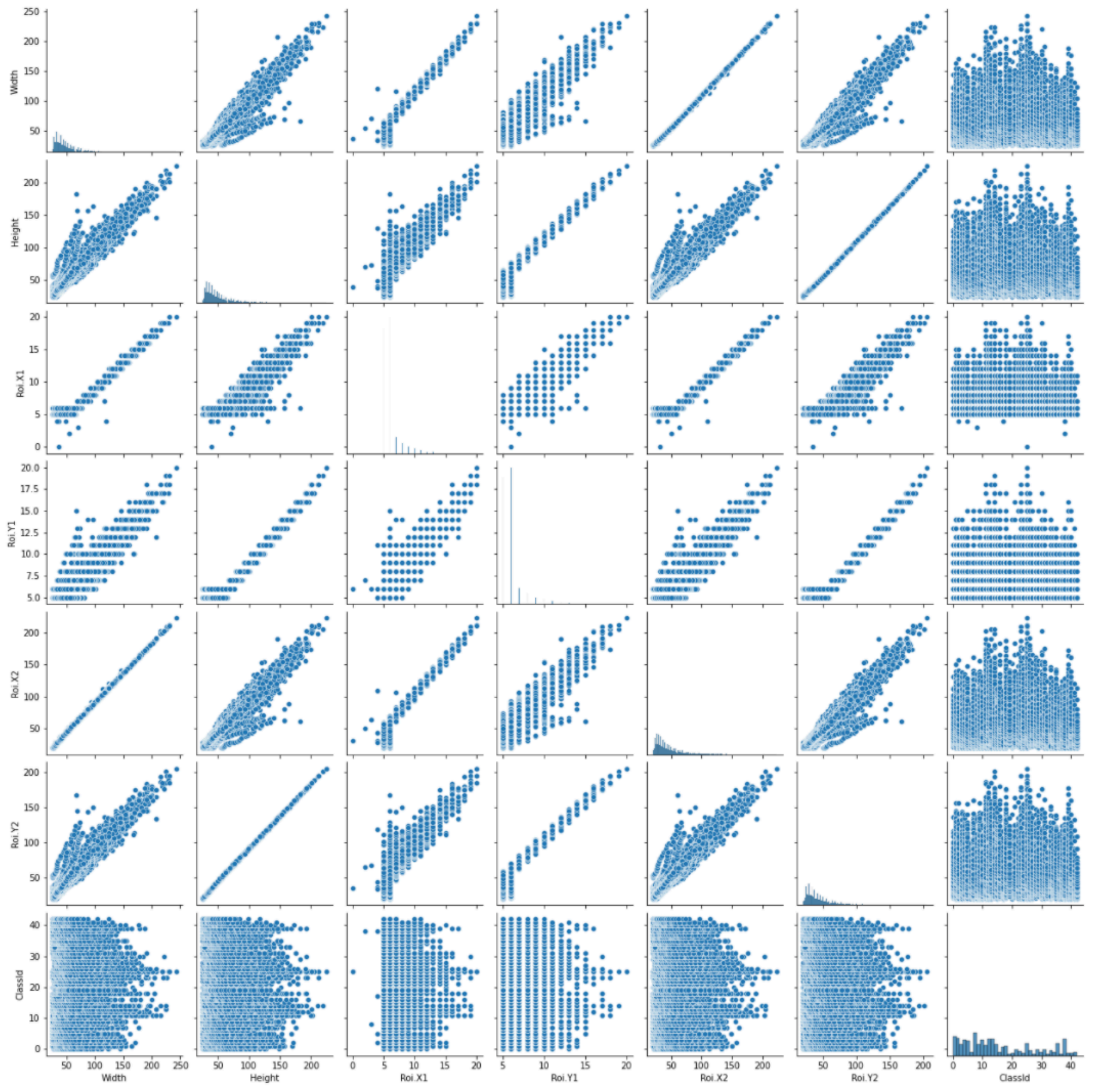
To this end, I have used the Traffic Sign Dataset available on Kaggle to create a model which has a high classification accuracy of road signs. The dataset contains 43 images of road sign classes and nearly 50,000 images to work with. The main data file contains a path to the image and the corresponding shape of the sign (triangular, circular, etc.), the color of the sign (red, yellow, etc.), the image class ID and the sign ID. Using this dataset, I was able to create a model with an accuracy of almost 96%. When an image is fed into the model, it predicts the type of road sign that is in the image.

Data Wrangling

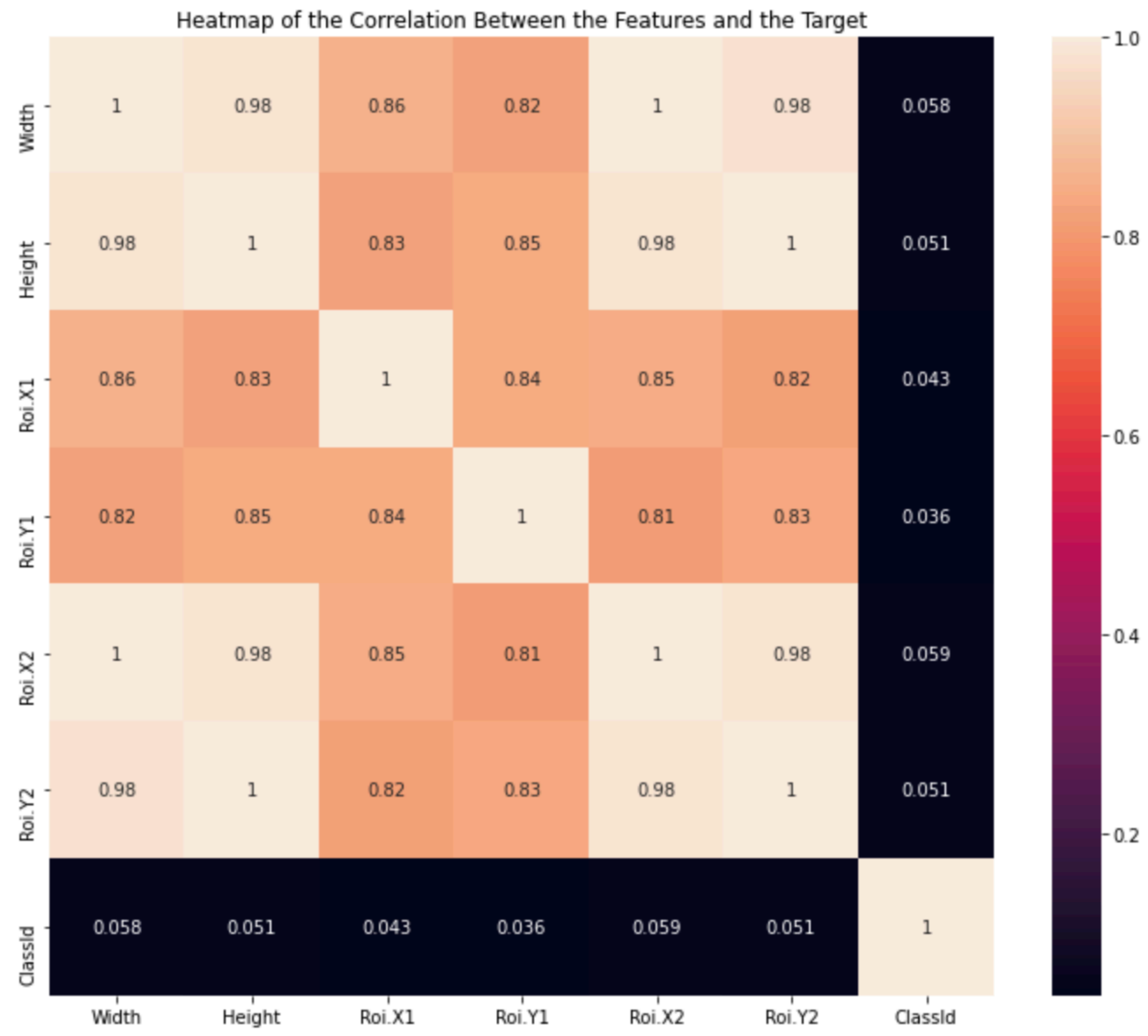
The raw data had a few features. These features were not correlated with each other because this contained pixel information about the images. Since the training and testing data had images, there was no particular wrangling of data needed. The data was very clean with no missing values in any of the rows. All the data types in the features were of the correct type. Hence there were no changes needed on that front. All in all, it was a very straightforward dataset in the sense that there were no major changes needed from me.

Exploratory Data Analysis

In my analysis, I wanted to look at all the features and try to see which ones are correlated to the others and to what degree. The easiest way for me to do this was to take a quick look at the “pairplot” feature of seaborn. This is shown below:



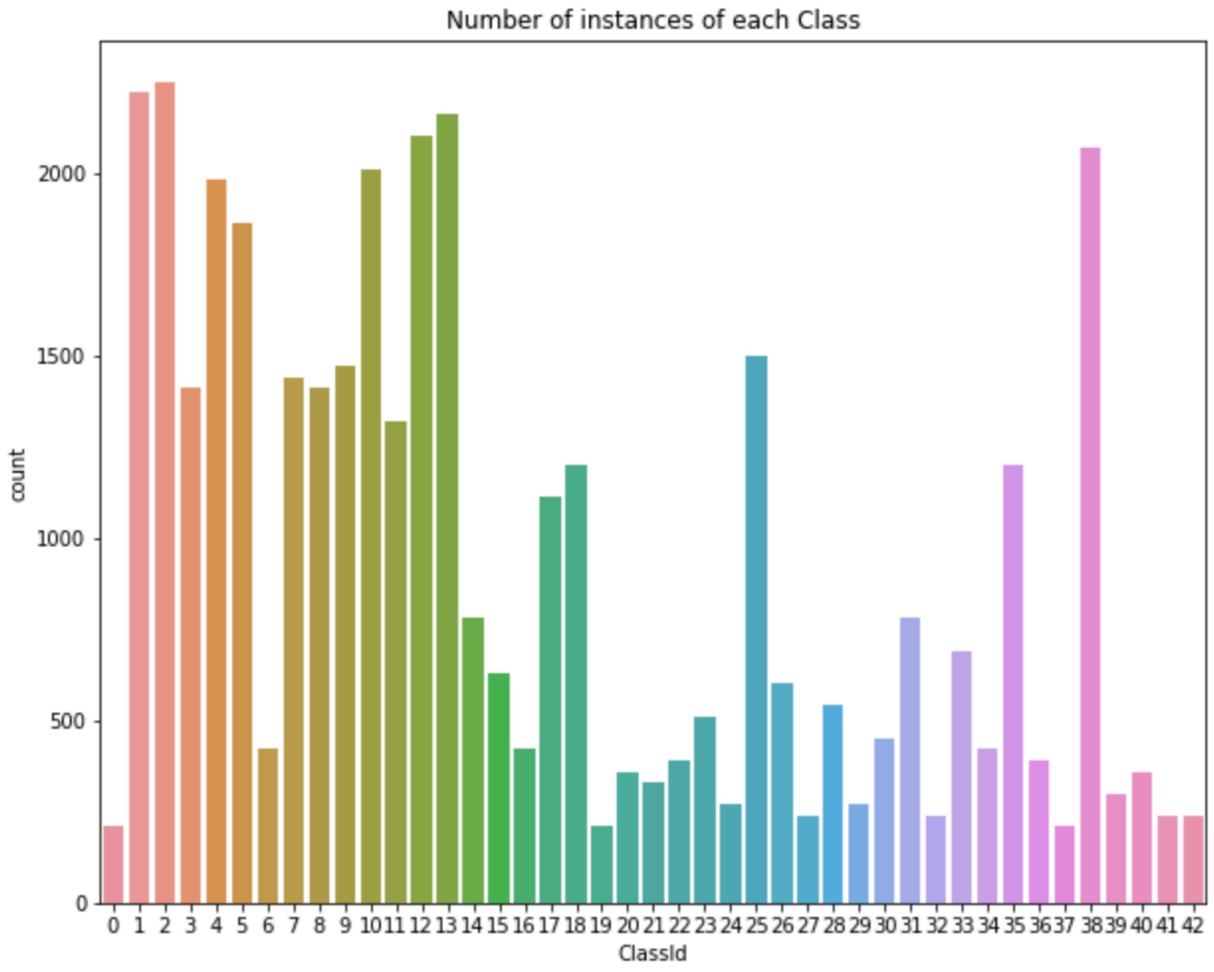
This is a very good way to look at the distributions of all the features, as well as their correlation with each other (demonstrated by the scatter plots). In this plot, we can see correlation between many columns (features). To take a closer look at the correlations, I plotted a heatmap, as shown below:



	Width	Height	Roi.X1	Roi.Y1	Roi.X2	Roi.Y2	ClassId
Width	1	0.98	0.86	0.82	1	0.98	0.058
Height	0.98	1	0.83	0.85	0.98	1	0.051
Roi.X1	0.86	0.83	1	0.84	0.85	0.82	0.043
Roi.Y1	0.82	0.85	0.84	1	0.81	0.83	0.036
Roi.X2	1	0.98	0.85	0.81	1	0.98	0.059
Roi.Y2	0.98	1	0.82	0.83	0.98	1	0.051
ClassId	0.058	0.051	0.043	0.036	0.059	0.051	1

Here, the independent variables seem highly correlated with one another, but not correlated at all with the target variable (Class ID). This is in agreement with the earlier mentioned fact that the columns mainly correspond to the images and the pixel information, which doesn't decide the class of the traffic sign.

Moving on, I wanted to see what the distribution of the classes in our training dataset looks like. I wanted to be certain that there are no major class imbalances in the data. Any major class imbalance would require some additional pre-processing. Here are the results:



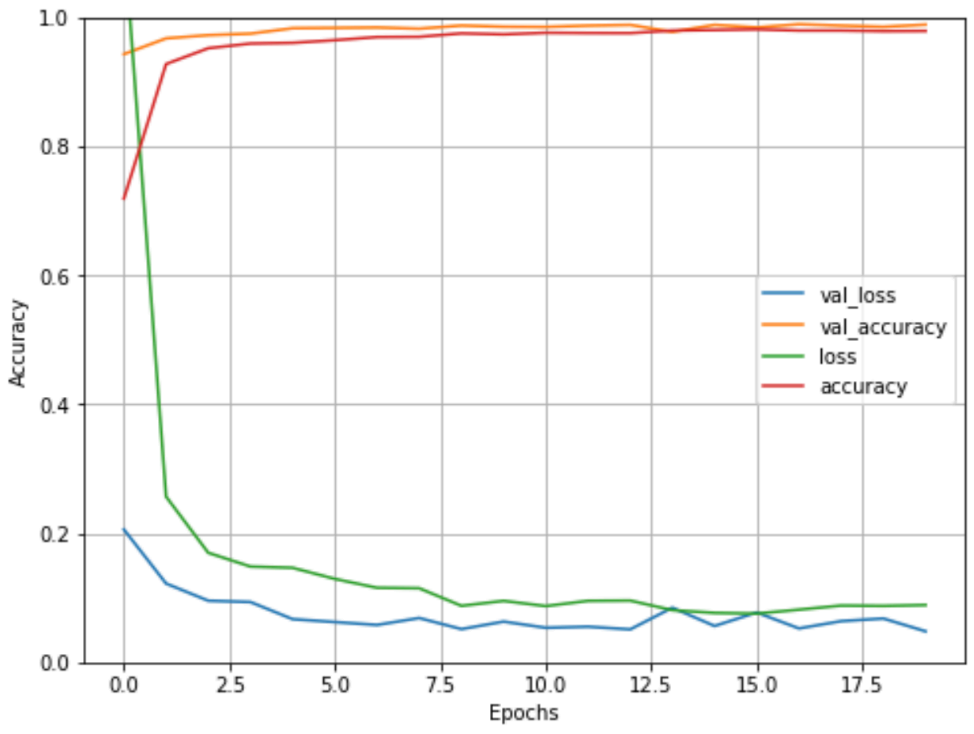
There are no major class imbalances. Most of the labels seem well-distributed. No further processing was required in this regard.

Data Preprocessing

Data pre-processing is a very important step in the model development. To work with the images as input, we needed to first convert the images into numpy arrays. After doing this, I assigned each array from each image to the corresponding label. This was followed by splitting the dataset into training and testing sets followed by encoding the categorical variables. This prepared the data to for the model to fit it (training) followed by the predictions (testing).

Modeling and Model Evaluation

For this part, I built a neural network with an input layer, 3 hidden layers and an output layer with 43 nodes. After compiling the model I fit it to the training dataset with the testing data as one of the inputs. This model showed a validation accuracy of almost 98%. Given how impressive the results were, I followed it up with taking a deeper look into in. This entailed evaluating the model. The following image shows the validation accuracy, validation loss, training loss and training accuracy with relation to the epochs:



This was followed by loading the test data, converting the test data to numpy arrays and predicting the label using our trained neural network. The training accuracy for the model was 96%.

The image above shows the actual and predicted class for different images in the test dataset.

Key Takeaways

One of the main takeaways for me in this capstone is development of a neural network model. While the examples in the Springboard curriculum prepare you for a basic neural network and the examples online are fairly straightforward, this is not always the case when you are defining your own model. Your model highly depends on the kind of input you have. The layer can be ‘dense’ or ‘conv2D’ or any other kind depending on the input dimensions. Some research and some messing around with the type of layers and the parameters such as the activation function, filters, kernel size, input dimension, number of nodes, etc.

The accuracy of the model, even with some very dark images was about 96%. This proves the efficacy of a neural network in processing images and classifying them, even if there are a lot of classes to predict (43 in our case).

Future Research

As always, more the data, the better. If we can somehow gather more images of the traffic road signs, we will probably be able to build a better model. I must mention that I mean increasing the *number of data points* as opposed to increasing the number of features. The model is already pretty accurate, but when it comes to applications such as a self-driving car, a 96% accuracy won't make the cut. In fact, for such applications, the accuracy would ideally be 100% to prevent any mishaps as it is a matter of life and death for the driver/passengers. More data and better model development will help in this application.