# Machine Learning Engineer Nanodegree

## Capstone Proposal

Aditya Gupta
July 16th, 2017

## Using Deep Learning to Identify Traffic Signs

### Domain Background

Self-driving cars are the most fascinating and innovative technology of the coming generation and its place among current vehicles and roads is dependent on its safety. Autonomous vehicles should be smart enough to avoid accidents and follow the traffic rules while maintaining reliability. One of the things to improve safety is to improve the ability of the car to detect and understand traffic signs while driving. This task can be divided into 2 separate problems of detection and classification. A large number of papers have been written on the former problem and solutions include using a combination of Color Normalization, Edge Detection, Contouring, etc. This project will focus on the latter problem by applying deep learning for classifying the Traffic Signs.

My motivation for doing this project is the experience that often when we are on vacation or a trip to any other country, the traffic signs vary and we may not necessarily know what they mean. In this project, I aim to use the Deep Learning model on android as a quick way to identify these traffic signs.

### Problem Statement

Safety is a big concern when handing over control of our vehicle to AI. Here we look at the problem of identifying Traffic Signs which would allow the autonomous vehicle to exercise caution and follow traffic safety regulations. One of the common methods for doing this task is to use the image's HOG features to detect shapes. ( Jack Greenhalgh ) Here I will build a Deep Learning model to perform the classification.

## Datasets and Inputs

The dataset I will use for training and evaluating my model is the German Traffic Sign Recognition Benchmark available here ( Dataset )(*cite*). Once the model is ready and implemented on android, the user will provide an input image for classification. The training data set contains 39,209 training images in 43 classes while the test dataset contains 12,630 images.

Each image is a cropped photo of a traffic sign and each image contains only 1 traffic sign. The images are stored in .ppm format and have varying sizes. The dataset is divided into a number of directories with 1 directory for each class. Each directory also contains a .CVS file for annotations. The images within each directory a divided into tracks and each track contains 30 images of one physical traffic sign.

## Solution Statement

As mentioned earlier, A model based on deep learning will be trained to identify the traffic signs. I plan to use a combination of multiple Convolution Layers, Pooling Layers, Fully connected layers, etc. in the model. This will allow the model to have enough parameters to pick up the structure of the traffic signs and classify them. After the various parameters are trained, the model will be transferred to android for making predictions.

## Benchmark Model

The dataset being used has been taken from a previous competition (German Traffic Sign Recognition Benchmark 2011) and the winners have been listed here. I will use the 4th position Method of Random Forests by team CAOR (96.14 % accuracy) as the benchmark model as it will help us compare a non-Deep Learning solution with our Deep Learning Model.

n this section, provide the details for a benchmark model or result that relates to the domain, problem statement, and intended solution. Ideally, the benchmark model or result contextualizes existing methods or known information in the domain and problem given, which could then be objectively compared to the solution. Describe how the benchmark model or result is measurable (can be measured by some metric and clearly observed) with thorough detail.

## Evaluation Metrics

Since we are training a classifier, we have the option of picking Accuracy, Recall, Precision or F-score as the metric for evaluating our model performance. F-score is useful when the number of samples belonging to one class is much more than the other. However, in our case the number of samples for each class are fairly evenly distributed and so we can use the Accuracy as an appropriate evaluation metric.

Also, the benchmark model has specified an accuracy and it makes sense to also judge our model performance by accuracy so that the two models can be compared fairly.

## Project Design

The first step will be to obtain the data for training and evaluating our model. This can be obtained from the link provided in the Dataset Section above. After the data has been obtained, we see that the images are of varying sizes. To apply CNNs efficiently we need to resize the images to a fixed size which is neither too small nor too big. If the image is too big then the training time and memory requirement will increase significantly, and if the image size is too small, it may not contain enough information for image to be classified properly. After resizing, the images will also need to be normalized in terms of values. This will not change the image but will help prevent the outputs being squashed from activation functions in the layers. The resized and normalized images can then be used for training.

After resizing, the images are still in separate files and for efficient training we need to combine them into a single file. Therefore, before feeding the data to the Neural Network it needs to be merged together and randomly shuffled because the data is initially ordered by class. If the data is ordered, it will cause the learner to be biased towards one class more than the other depending on the order of inputs while training.

After data processing, I will begin to implement the Deep Learning network. It will consist of multiple Convolution layers, pooling layers, Fully connected layers, etc. I will also apply techniques like dropout to improve the robustness of the model.

I will then train the model and continue to improve it, by varying the number and type of layers and their parameters till a satisfactory performance is achieved. After that I will begin to implement the android application by transferring the trained model and creating the User interface for selecting an image to be classified by the model. The selected image will have to be resized so that it can be fed into the model.