# Udacity's Machine Learning Engineer Nanodegreee Program
# State Farm Distracted Driver Detection
# Capstone Proposal

Rodrigo S. Veiga

May 2, 2018

This proposal is based on the Kaggle's homonymous challenge.

## 1 Domain Background

We have all been there: a light turns green and the car in front of you does not budge. Or, a previously unremarkable vehicle suddenly slows and starts swerving from side-to-side. When you pass the offending driver, what do you expect to see? You certainly are not surprised when you spot a driver who is texting, seemingly enraptured by social media, or in a lively hand-held conversation on their phone. According to the CDC motor vehicle safety division, one in five car accidents is caused by a distracted driver. Sadly, this translates to 425,000 people injured and 3,000 people killed by distracted driving every year. State Farm hopes to improve these alarming statistics, and better insure their customers, by testing whether dashboard cameras can automatically detect drivers engaging in distracted behaviors.

## 2 Problem Statement

Given a dataset of 2D dashboard camera images, the idea is to develop an machine learning algorithm to automatically classify driver's behavior. This task can be achieved by the extraction of relevant information stored in the training set, which is composed by labeled images (each label corresponds to a action the driver was doing when the picture was taken). The algorithm must learn the most important features from the training data and check its prediction performance in other images which are not part of the training set.

Are they driving attentively, wearing their seatbelt, or taking a self with their friends in the backseat? Such an algorithm could be highly important to prevent accidents if it could automatically warn distracted drivers based on their behavior captured by on-board cameras.

## 3 Datasets and Inputs

The dataset is composed by driver images (training and test sets), each taken in a car with a driver doing something in the car (texting, eating, talking on the phone, makeup, reaching behind, etc).

There ten classes to predict are:

- c0: safe driving
- c1: texting - right
- c2: talking on the phone - right
- c3: texting - left
- c4: talking on the phone - left
- c5: operating the radio
- c6: drinking
- c7: reaching behind
- c8: hair and makeup
- c9: talking to passenger

The data files can be found here.

Both training and test sets images are going to be transformed in four dimension tensors to fed in Keras. Particularly, in order to improve performance, the test set will be shuffled and part of it will be retrieved as a validation set.

**File descriptions**

- `imgs.zip`: zipped folder of all (train/test) images.
- `sample_submission.csv`: a sample submission file in the correct format.
- `driver_imgs_list.csv`: a list of training images, their subject (driver) `id`, and class `id`.

## 4 Solution Statement

Following the previous Dog Breed Prediction Project, a multi-layer convolutional neural network (CNN) will be implemented with Keras. The optimization will be made by the minimization of the multi-class logarithmic loss described below.

## 5 Evaluation Metrics

Following the challenge by Kaggle the models will be evaluated using the multi-class logarithmic loss (cross-entropy). Each image has been labeled with one true class. For each image, a set of predicted probabilities (one for every image) will be presented. The formula is then,

$$\mathcal{L}_{\log} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{m} y_{ij}\log(p_{ij}) \ ,$$

where $N$ is the number of images in the test set, $M$ is the number of image class labels, log is the natural logarithm, $y_{ij}$ is 1 if observation $i$ belongs to class $j$ and 0 otherwise, and $p_{ij}$ is the predicted probability that observation $i$ belongs to class $j$.

The presented probabilities for a given image are not required to sum to one because they are rescaled prior to being scored (each row is divided by the row sum). In order to avoid the extremes of the logarithm function, predicted probabilities are replaced with $\max(\min(p, 1-10^{-15}), 10^{-15})$.

The logarithmic loss quantifies the accuracy of a classifier by penalizing false classifications. Minimizing the logarithmic loss is basically equivalent to maximizing the accuracy of the classifier.

There are two main properties one would intuitively expects to interpret the cross-entropy as a cost function. Firstly, it is clear that $\mathcal{L}_{\log} \geq 0$. Secondly, if the predicted probabilities are close

to the desired output for almost all training inputs, the cost $\mathcal{L}_{\log}$ will be close to zero. However, these properties are also satisfied by more usual choices, like the quadratic cost. The advantage of the multi-class logarithmic loss function is that it avoids the problem of learning slowing down caused by the derivative of the sigmoid function in the quadratic cost.

Additionally, logarithmic loss heavily penalizes classifiers that are confident about an incorrect classification.

## 6 Benchmark Model

The hundred best placed models of the respective Kaggle's competition will be used as benchmark models. Although those results are calculated with approximately 69/100 of the test data, it seems reasonable to use these models as references, since they achieved the best performance in the competition. The champion accomplished a multi-class logarithm loss of 0.08739, while the tenth 0.14910. The fiftieth, in its turn, got 0.18954; the hundredth, 0.22594; among 1440 competitors.

## 7 Project Design

Since this problem is similar to the one presented in the Dog Breed Prediction Project it is reasonable to treat it with a CNN-based deep learning algorithm.

Initially, data exploration and data preprocessing will be carried out before the CNN implementation with Keras. After training the proposed architectures will be evaluated on test data using the evaluation metrics mentioned above.