Distracted driver detection using Machine Learning – Technical Report

The goal of this project is to detect distracted driver images of the set of input images with the help of machine learning. The need of the project is to ensure the driver and passengers safety and the attention of the driver while he/she drives. The future scope of the project is to alert the driver in real time whenever he/she loses focus while driving. It is like the airbags to safeguard passengers from accident with the help of car airbag system. The vehicle manufacturer can set a camera facing the driver with the preinstalled system with real time machine learning algorithm and alarms the driver whenever there is an abnormality.

The first doubt I had before the start is to select the language I am going to work on. After much consideration, I preferred Python over R because of the enriched libraries in Python. I chose to work with jupyter notebook in conda environment with the base of tensorflow and keras. I came across several algorithms for image classification and decided to go with Convolutional neural networks (CNN). The process started with the data pre-processing method which includes the preparation of images with the necessary folder structure according to the libraries I am going to use. Initially, I planned the whole project differently with k-fold cross validation. With sklearn and stratified kfold, I tried to fit my CNN model. I used 11 splits since we have 10 classes, but the process could not cross one iteration since I used three layers with 1024, 512 and output layer is 10 for 10 different classes. Since it cannot afford the (73728*1024) trainable parameters after one iteration due to OOM (Out of memory) error, I was forced to change my code to normal training-validation with one iteration.

I designed a CNN model with necessary batch normalization, max pooling, flatten and dropout. This model was purely a trial and error until I got good results. I compiled the model using the adam optimizer, and categorical cross entropy loss. I used image generator to get my images using flow from directory. I designed that image generator to split the data as 80:20 being training and validation data with the target size of 100*100. I have used image augmentation with horizontal flip, rotation, width, and height shift to generalize the data to avoid over fitting. I fit the model with 100 epochs with steps per epoch being the length of training set and validation steps being the length of validation set with the batch size of 64. After 100 epochs, I got more than 97% training accuracy and more than 96% validation accuracy with the training loss is less than 0.1 and validation loss is 0.144. Then I wanted to show the way the model has been trained with two plots – one between training and validation accuracy and another between training and validation loss.

Then I created another image generator for test images with a rescale. The test image size is same as training and validation image size. Then, I predict the model using predict function. With the prediction, I restructured the data to a data frame. I took the base model of data frame with sample submission csv and changed the values of the csv according to the predictions value. Then I did exploratory data analysis with the training data to represent the number of images in each class with a bar chart and another bar chart with given test data for comparison. Upon completion of data analysis, I converted the data frame to csv format.