STATE FARM DISTRACTED DRIVER DETECTION
UDACITY MACHINE LEARNING CAPSTONE PROJECT

12/5/19

OVERVIEW

1. Domain Background

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. It is very important

Me and my wife love doing road trips and spend a lot of time traveling on the road. At few occasions we have also witnessed how distracted drivers/driving could be dangerous to not only the drivers themselves but to the people around as well.

2. Problem Statement

For my Capstone project I will develop a Machine/Deep learning agent to address the challenge presented by the State Farm on the Kaggle to predict the behavior of the driver from the image provided by the dashboard camera of the car.

Given a dataset of 2D dashboard camera images, our aim is to classify the behavior of the driver in the image and predict whether they are driving attentively, wearing their seatbelt,   taking a selfie with their friends in the backseat, or involved in any other distracted behavior. This program can be used to alert the drivers whenever they are getting engaged into any distraction while driving.

3. Datasets and Inputs

Datasets are provided by the State farm for this problem and can be found on Kaggle at the given link - https://www.kaggle.com/c/state-farm-distracted-driver-detection/data

driver_imgs_list.csv.zip - A list of training images, their subject (driver) id, and class id -

imgs.zip - zipped folder of all (train/test) images

sample_submission.csv.zip - A sample submission file in the correct format

Input is in the form of driver images, each taken in a car with a driver doing something in the car (texting, eating, talking on the phone, makeup, reaching behind, etc) and the goal is to predict the likelihood of what the driver is doing in each picture.

The 10 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

4. Solution Statement

A good solution for this problem would be to build a program/agent which can correctly predict the distracted behavior of the driver and alert the driver timely and appropriately to prevent any unfavorable event.

Looking at the nature of problem, I believe a deep learning algorithm based on CNN might be a right choice to solve it. Model will be trained using the training data and then driver's behavior will be predicted for the test images. Based on the prediction, model can be evaluated. The CNN model implemented in TensorFlow/Keras can be then optimized to minimize the multi-class logarithmic loss, which will be used to evaluate the performance of the program.

5. Benchmark Model

For Benchmarking, the model can be compared against the top model from the Public Leaderboard for this competition and the goal for this project would be to gain the rank in top 50% of the public leaderboard.

6. Evaluation Metrics

Submissions are evaluated using the multi-class logarithmic loss. Each image has been labeled with one true class. For each image, you must submit a set of predicted probabilities (one for every image). The formula is then,

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij}),$$

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

The submitted 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 log function, predicted probabilities are replaced with max(min(p,1−10−15),10−15).

7. Project Design

First step in the project implementation would be to load the data/images provided as input. The data would then be required to analyzed and pre-processed before using it to train the model.

Once the data is pre-processed to be in the format we want, we can implement the Machine/Deep Learning model. For this specific problem I think a CNN model implemented in Keras can give us good results. Once the model is created we will compile it and train it against the training data. We may use the model checkpointing to save the model that attains the best validation loss.

Depending on the performance of the model, we may implement more techniques or fine tune the parameters to improve the performance. One of the technique which don't necessarily improve the performance but save a lot of training time without compromising with the performance is Transfer Learning. Once we have the optimized model we can use it against the test data to predict the behavior of the distracted driver and save the predicted probabilities in a file as per the format required for the submission on the Kaggle.