Proposal: State Farm Distracted Driver Detection

Domain Background

The domain of this problem is computer vision. Computer vision is a branch of machine learning concerned with the automatic extraction, analysis and understanding of useful information from a single image or a sequence of images (BMVA, n.d.). Computer vision began in the 1960's, when a person named Larry Roberts wrote his PhD thesis on the possibility of extracting 3D details and information from 2D images(T.S. Huang, n.d.). In the 70's, some progress was made on the interpretation of 2d images to 3d images (Hari Narayanan, et al, n.d.). In the 80's, optical character recognition systems that recognize letters, symbols and numbers were used in several industries (Quick history, n.d.). In the 90's, new applications of computer vision were possible as computers became more powerful and common (Quick history, n.d.). In the 2000's, computer vision was used to process large datasets, videos and could understand motion, patterns and predict outcomes (Hari Narayanan, et al, n.d.). This problem interests me because I want to have a better understanding of computer vision classification problems specifically human actions and the different approaches I could use in order to solve it.

Problem Statement

The problem that we are trying to solve is a multi-class classification problem. We are tasked to properly predict and classify driver's behavior given the dashboard images of people doing 10 different actions, 9 of which are considered actions of distracted behavior. The 10 classes are as follows: 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

Motor vehicles are defined as automotive vehicles not operated on rails (Motor Vehicle,n.d.). The automotive industry sparked and began production in the second half of the 1800's with the introduction of vehicles that provide a quick and efficient way of going from point A to point B. In 1901, the first modern motorcar was designed and created (History.com, 2010). The first documented road traffic fatality is in 1869, when a woman was killed after she fell under the wheels of an experimental steam car built by her cousins **(**Offaly History, 2007**)**. Since then, It has always been important for drivers to pay attention on the road but people who get distracted while driving endanger not only themselves but others as well. Who are distracted drivers? These are drivers who have their attention taken away by anything, be it a cellphone, food or navigation systems (Distracted Driving, 2017). Distractions while driving has become an increasingly concerning issue as its been reported that 1.3 million people die in road crashes each year and about 20-50 million people get injured each year (U.S. Department of Transportation, n.d.). About 90% of all road crashes are caused by preventable driver behavior (U.S. Department of Transportation, n.d.). Road traffic injures are projected to be the 5th leading cause of death worldwide by 2030 unless a solution is put in place(Annual Global Road Crash Statistics, n.d.). A solution to this problem is using machine learning computer vision models to classify driver actions with the end goal of determining whether the drivers are engaging in distracted behavior or not.

Datasets and Inputs

The dataset being considered is the one provided in the Kaggle competition. It contains 2 folders, one which contains the training images and the other which contains the test images. There are approximately 2,500 images for each of the 10 classes in the train folder and 102,149 images for the test data. All images are colored and 640 x 480 in size. The images capture the driver from a side-view dashboard camera. I will make use of the training images to train a model and test images to test the model.

Note* If my GPU(GTX 1060 6GB) takes incredibly long to train then, I will opt to use only a subset of the data.

Sample image:

Solution Statement

For this computer vision multi-class classification problem, working with Convolutional Neural Networks would be a good idea because CNN's are good at classifying images and are known to give results with high accuracy. Keras application models with their pre-trained weights could reduce the time it takes to train while still yielding good results. But before we do that, pre-processing the images will be necessary. Reducing image size and dividing the images into the RGB channels could make processing of the images more manageable. Once the model is fit, we will need to predict the labels of the test set to determine which of 10 categories each picture belongs to. The results, evaluation metric and benchmark model will then tell us if/where we still need to improve the model.

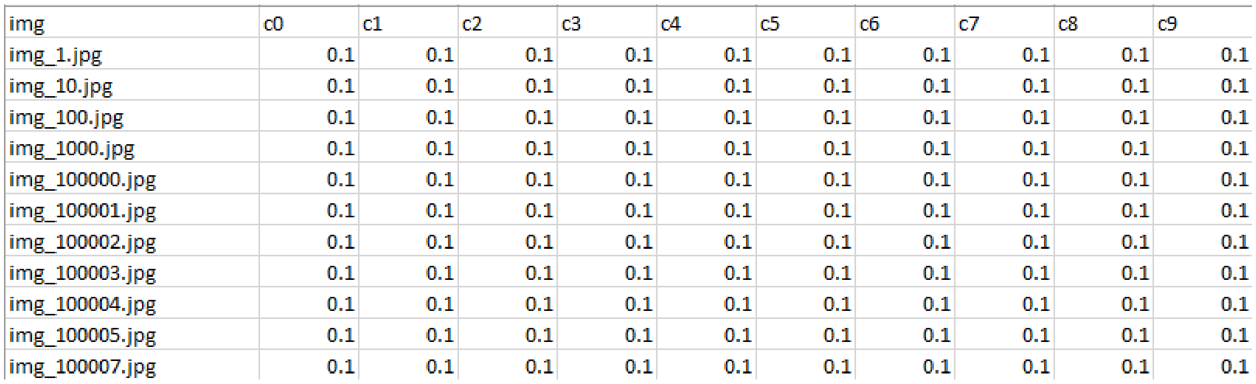
Benchmark Model

I will base the benchmark results on the Kaggle public leaderboard as I was unable to find a benchmark model as such. Since everyone in the leaderboard must follow the same rules and evaluation metric, it makes it good for benchmarking. My target result for this project is to reach the top 10% (≤ 144 of 1440) people.

Evaluation Metrics

I will make use of the Multi-class logarithmic loss. This metric is used in many computer vision classification problems because it measures the accuracy of a classifier by penalizing false classifications. It is also a good metric for this problem because in order to calculate log-loss, the classifier must assign a probability to each class rather than yielding the most likely class**()**. If we know the distribution of response values, we could use this information to improve our model and dataset. In our problem, the accuracy is very important, so we will need the information. Using other metrics such as classification accuracy, only provides us with the accuracy. We need to compare our results to a benchmark, so it is best to stick with the metric used by the Kaggle competition.

Sample output:



img	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9
img_1.jpg	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
img_10.jpg	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
img_100.jpg	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
img_1000.jpg	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
img_100000.jpg	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
img_100001.jpg	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
img_100002.jpg	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
img_100003.jpg	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
img_100004.jpg	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
img_100005.jpg	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
img_100007.jpg	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

Multi class log loss example:

```
>LogLossMulti (["bam", "ham", "spam"], [[1, 0, 0], [0, 1, 0], [0, 0, 1]])

[1] 2.1094237467877998e-15

>LogLossMulti (["bam", "ham", "spam"], [[0, 0, 1], [1, 0, 0], [0, 1, 0]])

[1] 34.538776394910684
```

```
>LogLossMulti (["bam", "ham", "spam", "spam"], [[0.8, 0.1, 0.1], [0.3, 0.6, 0.1], [0.
15, 0.15, 0.7], [0.05, 0.05, 0.9]])

[1] 0.2990
```

Retrieved from Mark Needham,first steps with log loss
http://www.markhneedham.com/blog/2016/09/14/scikit-learn-first-steps-with-log_loss/

From this example, we can see that when the prediction is completely the same as the actual value, the log loss results to a 2.11e-15 and when the prediction is completely wrong, the log loss reaches 34.54.

Project Design

A solution I have thought of is, step 1, input the training data. Step 2, use a subset of the data to train. Step 3, Reduce and scale images from the dataset to a more manageable size. Step 4, augment the dataset by adding images with noise and rotation. Step 5, make the CNN architecture but most optimally make use of the keras application and pre-trained model weights such as Xception, VGG-16, Resnet50. Step 6, I will fit the model and save it. Step 7, I will then test with a subset of the test dataset. Step 8, validate results/accuracy. Step 9, Plot the train and test dataset against the fitted values/epoch to see how much the model is overfitting. Step 10, Adjust architecture, model and parameters. Step 11, Repeat until it meets target results. Step 12, Make use of the complete training dataset and 31% of the test dataset just as the Kaggle public leaderboard. Step 13, make final adjustments.

Despite the (optimal) solution I have thought of above, I am going to consider working with bag of words because it could be useful if I could tell when a “driver is holding a cup”. Another approach I am considering is face and hand recognition, this could work to see if the driver is distracted but it may be punished hard if its unable to detect what exactly he is distracted from. Using multiple models is another option that could achieve great results, but it is a resource expensive solution. Once I am done with the model, I could try making a web app that shows the test results visually with selected images, maybe I could try with my own images.

Sources:

Motor Vehicle. (n.d.). In Merriam Webster’s online dictionary. Retrieved October 25, 2017, from https://www.merriam-webster.com/dictionary/motor%20vehicle

Distracted Driving. (2017). In Centers for Disease Control and Prevention. Retrieved October 25, 2017, from https://www.cdc.gov/motorvehiclesafety/distracted_driving/

Road Traffic Injuries. (2017). In World Health Organization. Retrieved October 25, 2017, from http://www.who.int/mediacentre/factsheets/fs358/en/

Distracted driving global fact sheet. (n.d.). In U.S. Department of Transportation National Highway Traffic Safety Administration. Retrieved October 25, 2017, from http://usdotblog.typepad.com/files/6983_distracteddrivingfs_5-17_v2.pdf

Annual Global Road Crash Statistics. (n.d.). In Association for safe international road travel. Retrieved October 25, 2017, from http://asirt.org/initiatives/informing-road-users/road-safety-facts/road-crash-statistics

Ward, Marry. (2007). In Offaly History. Retrieved October 25, 2017, from http://www.offalyhistory.com/reading-resources/history/famous-offaly-people/mary-ward-1827-1869

Automobile History. (2010). In History. Retrieved October 25, 2017, from http://www.history.com/topics/automobiles

Texting and Driving Accident Statistics. (n.d.). In Edgar Snyder. Retrieved October 25, 2017, from https://www.edgarsnyder.com/car-accident/cause-of-accident/cell-phone/cell-phone-statistics.html

Andrew B. Collier(2015)Making sense of Logarithmic Loss. In Exegetic. Retrieved Octover 26,2017 from http://www.exegetic.biz/blog/2015/12/making-sense-logarithmic-loss/

State Farm Distracted Driver Dataset. (2016). In Kaggle. Retrieved October 25, 2017, from https://www.kaggle.com/c/state-farm-distracted-driver-detection/data

Ritchie Ng. (n.d.). Evaluating a classification model. Retrieved October 30,2017, from http://www.ritchieng.com/machine-learning-evaluate-classification-model/

What is computer vision? (n.d.). In BMVA. Retrieved October 30,2017, from http://www.bmva.org/visionoverview

Quick History of Machine Vision. (n.d.). In EPIC systems. Retrieved October 30,2017, from https://www.epicsysinc.com/blog/machine-vision-history

Hari Narayanan, Libin Sun, Greg Yauney, et al. (n.d.). Introduction to computer vision. Retrieved October 30,2017, from https://cs.brown.edu/courses/cs143/lectures/01.pdf

T. S. Huang. (n.d.) Computer vision: Evolution and Promise. Retrieved October 30,2017, from https://cds.cern.ch/record/400313/files/p21.pdf