A PROJECT REPORT
ON

DRIVER ACTIVITY DETECTION AND DISTRACTION ALERTNESS SYSTEM

BY

NAME	ROLL NO.
SRIYA SAINATH	115CS0239
SOUMYA GOURAB SAHOO	115CS0237

UNDER THE GUIDANCE OF
PROF. PABITRA MOHAN KHILAR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY, ROURKELA

January- April, 2018

Acknowledgements

We are profoundly grateful to **Prof. Pabitra Mohan Khilar** for his expert guidance and continuous encouragement throughout to see that this project achieves its target since its commencement to its completion.

We also express our sincere heartfelt gratitude to everyone who helped us directly or indirectly during this course of work.

Sriya Sainath
Soumya Gourab Sahoo

ABSTRACT

The risk of drivers engaging in distracting activities is increasing as in-vehicle technology and carried-in devices become increasingly common and advanced. Consequently, distraction and inattention contribute to crash risk and are likely to have an increasing influence on driving safety.

In an attempt to mitigate these alarming situations, this paper explores using a dashboard camera using deep learning approach to automatically detect distracted drivers. The method consists of 3 convolutional layers of network with total 32 filters with size of 3x3, each of which is followed by a max-pooling layer size of 2x2, and 2 fully-connected dense layers. This method of detecting and alarming a driver whose concentration is off the road by training a Convolutional Neural Network (CNN) from the scratch yielded a high accuracy level of 98.54%.

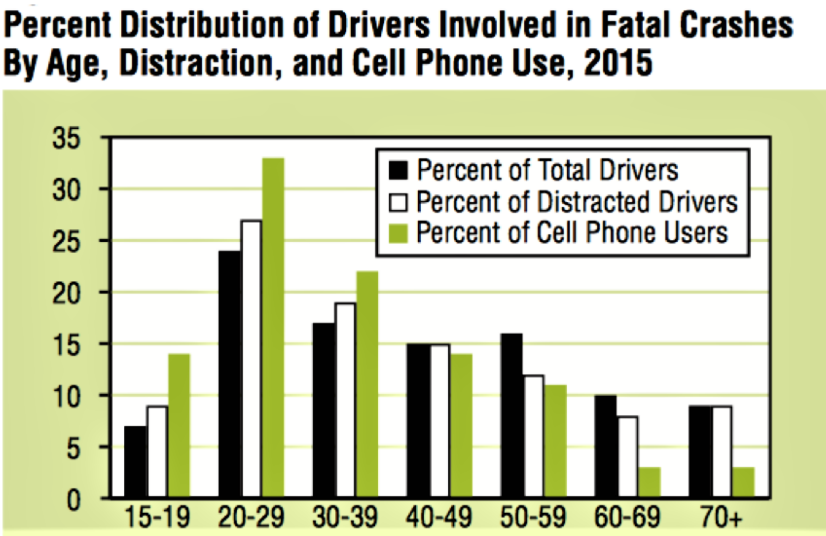
Keywords: Distracted Driver, Crash Risk, Deep Learning, Computer Vision, Convolutional Neural Network

Contents

Chapter 1

Introduction

Distracted driving is any activity that diverts one's attention from focusing on the road. Drivers, though supposed to be focused on driving by the law, very often resort to other activities while driving, including talking or texting on their phones, eating and drinking, talking to people in the vehicle, fiddling with the stereo, entertainment or navigation system anything that takes their attention away from the task of safe driving. These distractions can result in crashes/accidents. Proved study estimates that 31% of crashes are caused by a driver using his or her cell phone with financial losses of 3% GDP every year. Therefore, in India it can be estimated that the total loss due to road accidents is about 108 crores per day.



In order to alert the distracted driver and better ensure their safety and that of other commuters, we can design an alarm system that can detect the distracted behavior of car drivers by using a dashboard camera. This task is very meaningful for improving the drivers safety and can be easily applied to other applications such as triggering autonomous driving. This project primarily aims to build a computer vision system to detect and alarm the distracted driver, having long term goals of serving as a metric to judge a drivers ability by competent authority.

Chapter 2

Literature Survey

This is a widely addressed prolem across all countries and hence has been long worked on. Many top solutions used pre-trained CNN models and the most popular are VGG-16 and ResNet, which were state-of-the-art one or two years ago and had been improving [2][3]. Besides these models, two other decent ideas are: ensemble, K nearest neighbor s (KNN) and data augmentation.

First, because the test set size is about four times as large as the training set, it is easy to overfit. Creating an ensemble of models can reduce variance and alleviate this problem. Since the images are taken from a video clip, many images can be very similar or almost identical and they should be classified to the same class. Applying KNN can yield stable results. When KNN was used for either test or the training test (data augmentation) with different ideology and, the results were good in both cases.

The same problem was also attempted as described [4] in two approaches: training a small network from scratch and also use pre-trained VGG-19 model. In this attempt, all images were reduced to 224 224 from 640 480. This could detract from performance by throwing away some information, but shorten the computing time. The weakness of the solution is that they tried only single models (i.e. no ensemble) and there does not seem to have a significant amount of fine tuning done on the pre-trained VGG-19 model. Although very time consuming, trying to fine tune pre-trained models a few times is a good investment as it can make the pre-trained models more customised towards the specific problem.

Chapter 3

Methodology

3.1 Technology Required

3.1.1 Domains

- Image Processing
- Deep Learning
- Computer Vision
- Machine Learning

3.1.2 Software Used

- Deep Learning Library - Keras
- Tensorflow Backend
- Python and Opencv for Image processing
- Datasets from kaggle

3.2 Procedure Followed

Our designed system detects in real-time if a driver is conducting any activity that loses his/her focus while driving and triggers an alarm if necessary.
We utilized a technique known as transfer learning where a pre-trained network is used for initialization weights and then further trained to learn the idiosyncrasies of our data. In our net model, we perform global average pooling (GAP) just before the final output layer at the end. This helps the convolutional neural network to have localization ability despite being trained on images.
Our model "learnt" what distracted driving looks like from images in the data-set and accurately predicted and sorted unseen and new images to the following 10 classes:

- 0: safe driving
- 1: texting - right
- 2: talking on the phone - right
- 3: texting - left
- 4: talking on the phone - left
- 5: operating the radio
- 6: drinking
- 7: reaching behind
- 8: hair and makeup
- 9: talking to passenger

3.3 Code Snippet

Distracted Driver detection with 3 layer nework training :

```
model=Sequential()
model.add(Convolution2D(nb_filter=32, nb_row=3, nb_col=3, input_shape=(150,150,3),
    dim_ordering="tf"))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2),dim_ordering="tf"))
model.add(Convolution2D(32, 3, 3,dim_ordering="tf"))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2),dim_ordering="tf"))
model.add(Convolution2D(nb_filter=64, nb_row=3, nb_col=3,dim_ordering="tf"))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2),dim_ordering="tf"))
model.add(Flatten())
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(10))
model.add(Activation('softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adadelta', metrics=['accuracy'
    ])
 configuration for generating training data
train_datagen=ImageDataGenerator(rescale=1.0/255, shear_range=0.2, zoom_range=0.2,
    horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1.0/255)
model.save('driver_state_detection_small_CNN.h5')
```

After successful training on the model, it was able to distinguish any live or stored image clip to detect if the driver is attentive or distracted.

Chapter 4

Motivation and Objective

4.1 Motivation

Driver distraction is occurring with greater frequency in contemporary times. Consequently, distraction and inattention contribute to crash risk and are likely to have an increasing influence on driving safety. Challenges of detecting distractions at the crash site and reluctance of drivers to admit to being distracted are a limitation for this method of estimating the linkage between distraction and injuries and fatalities. A naturalistic driving study found that distraction and inattention contribute to approximately 80% of crashes or near crashes [4]. This increasing correlation between inattentive driving and crash risk is the major cause behind this project.

4.2 Objective

Hence, this cause for concern motivated us to develop and assess real-time distraction detection systems to guide technology development to enhance driver safety, and identify potential evaluation techniques to characterize and assess this emerging technology, serving as a warning/competency detection system.

Chapter 5

Results and Discussions

Driver Images Classified into Various Categories

The following results were concluded after training and predicting the data:

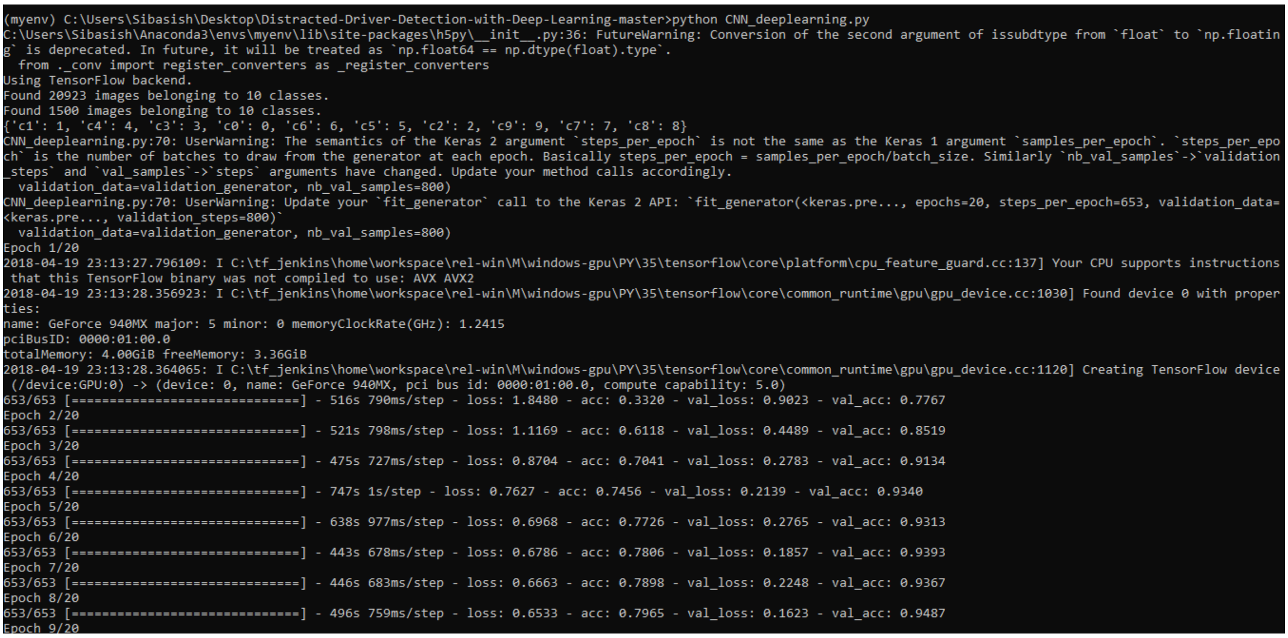
- Training and testing takes a substantial amount of the time consumed using the entire 4096 feature vector classification for 20 epochs.

- Regularization was successfully implemented using an additional hyperparameter to optimize prevent overfitting of data by limiting size of weights

- 96.54% accuracy and 0.56 logloss on cross validation was acquired while predicting the results.

- Many similar images taken within short time-frames make the data prone to over-fitting. Such classification challenges were successfully trained.

- To receive accurate test evaluations, cross-validation was required with 26 different persons in train set splitting the training set into 5 folds with 5 persons.

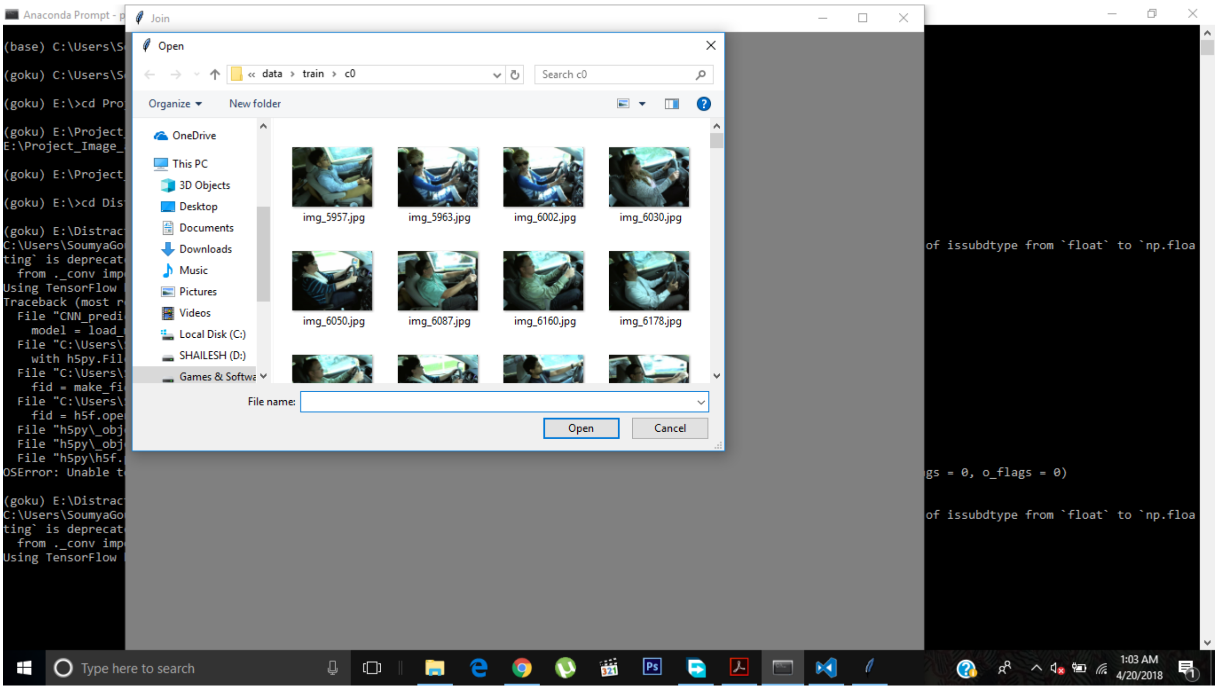


```
(myenv) C:\Users\Sibasish\Desktop\Distracted-Driver-Detection-with-Deep-Learning-master>python CNN_deeplearning.py
C:\Users\Sibasish\Anaconda3\envs\myenv\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion of the second argument of issubdtype from `float` to `np.floatin
g` is deprecated. In future, it will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
Found 20923 images belonging to 10 classes.
Found 1500 images belonging to 10 classes.
{'c1': 1, 'c4': 4, 'c3': 3, 'c0': 0, 'c6': 6, 'c5': 5, 'c2': 2, 'c9': 9, 'c7': 7, 'c8': 8}
CNN_deeplearning.py:70: UserWarning: The semantics of the Keras 2 argument `steps_per_epoch` is not the same as the Keras 1 argument `samples_per_epoch`. `steps_per_epo
ch` is the number of batches to draw from the generator at each epoch. Basically steps_per_epoch = samples_per_epoch/batch_size. Similarly `nb_val_samples`->`validation
_steps` and `val_samples`->`steps` arguments have changed. Update your method calls accordingly.
  validation_data=validation_generator, nb_val_samples=800)
CNN_deeplearning.py:70: UserWarning: Update your `fit_generator` call to the Keras 2 API: `fit_generator(<keras.pre..., epochs=20, steps_per_epoch=653, validation_data=
<keras.pre..., validation_steps=800)`
  validation_data=validation_generator, nb_val_samples=800)
Epoch 1/20
2018-04-19 23:13:27.796109: I C:\tf_jenkins\home\workspace\rel-win\M\windows-gpu\PY\35\tensorflow\core\platform\cpu_feature_guard.cc:137] Your CPU supports instructions
 that this TensorFlow binary was not compiled to use: AVX AVX2
2018-04-19 23:13:28.356923: I C:\tf_jenkins\home\workspace\rel-win\M\windows-gpu\PY\35\tensorflow\core\common_runtime\gpu\gpu_device.cc:1030] Found device 0 with proper
ties:
name: GeForce 940MX major: 5 minor: 0 memoryClockRate(GHz): 1.2415
pciBusID: 0000:01:00.0
totalMemory: 4.00GiB freeMemory: 3.36GiB
2018-04-19 23:13:28.364065: I C:\tf_jenkins\home\workspace\rel-win\M\windows-gpu\PY\35\tensorflow\core\common_runtime\gpu\gpu_device.cc:1120] Creating TensorFlow device
 (/device:GPU:0) -> (device: 0, name: GeForce 940MX, pci bus id: 0000:01:00.0, compute capability: 5.0)
653/653 [==============================] - 516s 790ms/step - loss: 1.8480 - acc: 0.3320 - val_loss: 0.9023 - val_acc: 0.7767
Epoch 2/20
653/653 [==============================] - 521s 798ms/step - loss: 1.1169 - acc: 0.6118 - val_loss: 0.4489 - val_acc: 0.8519
Epoch 3/20
653/653 [==============================] - 475s 727ms/step - loss: 0.8704 - acc: 0.7041 - val_loss: 0.2783 - val_acc: 0.9134
Epoch 4/20
653/653 [==============================] - 747s 1s/step - loss: 0.7627 - acc: 0.7456 - val_loss: 0.2139 - val_acc: 0.9340
Epoch 5/20
653/653 [==============================] - 638s 977ms/step - loss: 0.6968 - acc: 0.7726 - val_loss: 0.2765 - val_acc: 0.9313
Epoch 6/20
653/653 [==============================] - 443s 678ms/step - loss: 0.6786 - acc: 0.7806 - val_loss: 0.1857 - val_acc: 0.9393
Epoch 7/20
653/653 [==============================] - 446s 683ms/step - loss: 0.6663 - acc: 0.7898 - val_loss: 0.2248 - val_acc: 0.9367
Epoch 8/20
653/653 [==============================] - 496s 759ms/step - loss: 0.6533 - acc: 0.7965 - val_loss: 0.1623 - val_acc: 0.9487
Epoch 9/20
```

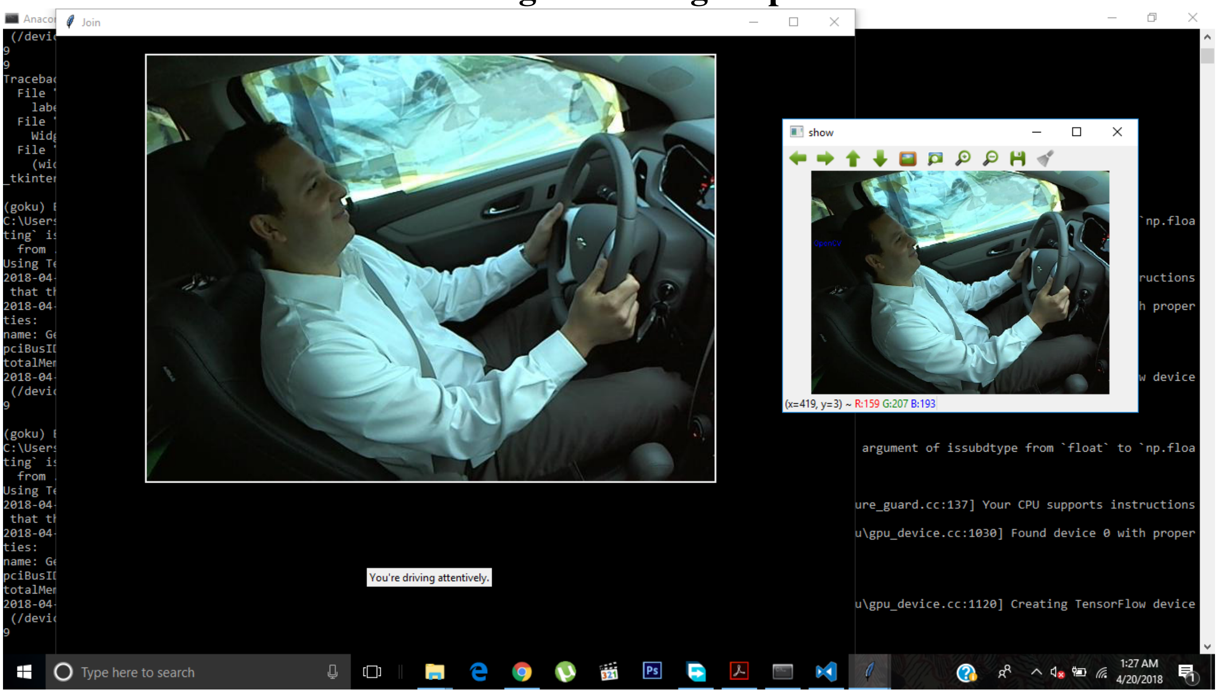
Model Training

- After the model was trained on the given huge data-set, the user was allowed to enter image of his choice through the interface provided.

- Using the trained data, the new image was classified into one of the given classes, alerting the user in case of distracted driving.



Allowing User Image Input



Displaying Results

- Thus, the model was able to successfully sort images according to their respective classes, and hence, fulfill the purpose of detecting inattentive driving.

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

The data-set that was used in this work that included around twenty thousand dashboard camera images for multiple drivers. These images were captured while the drivers were safely driving or were engaging in distracting behaviors (e.g. using a cell-phone for texting or calling, drinking, operating car accessories, etc.). After training the program, it was able to process any recorded images, whether real-time from the car dashboard or static images, and successfully categorize them under safe driving or alert what kind of distraction the driver is resorting to. Hence, it can be used as a warning system to prevent unsafe driving.

6.2 Future Scope

The current distraction detection system acts as a warning and suggestion measure. It can further be encompassed under legal terms if the alarm triggered when the driver is distracted is also used to intimate the traffic authorities. Hence, it can act as a means of determining a driver's competency and alerting the system against inattentive driving.

Further areas in which this work can be implemented are:

- Accident prevention
- Smart vehicles
- Driver alertness
- Automatic car driving assistance

References

[1] *Centers for Disease Control and Prevention. Distracted Driving. https://www.cdc.gov/motorvehiclesafety/distracted_driving/*

[2] *Simonyan, Karen, and Andrew Zisse rman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556(2014).*

[3] *He, Kaiming, et al. "Deep residual learning for image recognition." arXiv preprint arXiv:1512.03385 (2015)*

[4] *Singh, Diveesh. Using Convolutional Neural Networks to Perform Classification on State Farm Insurance Driver Images (2016)*

[5] *Hinton, Geoffrey E., et al. "Improving neural networks by preventing co-adaptation of feature detectors." arXiv preprint arXiv:1207.0580 (2012)*