DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DRIVER ACTIVITY RECOGNITION FOR VEHICLES USING DEEP CONVOLUTIONAL NEURAL NETWORK MODELS

Team Members:
Ravisankar MR 201601087
Naveen Kumar K 201601061
Ramasamy NS 201601085

Supervisor : Mr. R. Gnanavel
(Asst. Professor)
Mrs. G. Kalpana
(Asst. Professor)

DATE : 14-3-2020

ABSTRACT

- Driver decisions and behaviors are essential factors that can affect the driving safety. To understand the driver behaviors, a driver activities recognition system is designed based on the deep convolutional neural networks (CNN).

- Specifically, eleven common driving activities are identified, which are the normal driving, right mirror checking, rear mirror checking, left mirror checking, using in-vehicle radio device, texting, answering the mobile phone, Drinking or eating, Reaching behind, make-up, one handed driving respectively

- Among these activities, the first four are regarded as normal driving tasks, while the last seven are classified into the distraction group.

- The experimental images are collected using a low-cost camera, and ten drivers are involved in the naturalistic data collection. The raw images are segmented using the *Gaussian mixture* model to extract the driver body from the background before training the behaviour recognition CNN model.

- To reduce the training cost, transfer learning method is applied to fine tune the pre-trained CNN models.

INTRODUCTION

- Most accidents occur due to the driver being involved in the following 7 activities, [1]Using in-vehicle radio device, [2]texting, [3]answering the mobile phone, [4]drinking or eating, [5]reaching behind, [6]face make-up or hair adjusting, [7]one handed driving. These distractions can be briefly grouped into the three following major types,

  Visual - Taking eyes off the road. [1]

  Manual - Taking hands off the wheel. [2]

  Cognitive - Taking mind off while driving. [3]

- The experimental images are collected using a low-cost camera, where drivers are involved in the naturalistic data collection. The raw images are segmented using the *Gaussian mixture model* to extract the driver body from the background before training the behaviour recognition *Convolutional Neural Network* model.
- To reduce the training cost, *Transfer learning method* is applied to fine tune the pre-trained CNN models.

000

- *BENEFITS:* The detected driver's behavior is used to alert the driver. The system also automatically turns on the hazard lights and very slowly reduces the speed of the car or compels the driver to move to the side lane and stop the car to save the lives of the driver, passenger and the people travelling around the car.

LITERATURE REVIEW

PAPER	AUTHORS	DESCRIPTION
Driver activity recognition for Intelligent Vehicles: A deep learning approach	Yang Xing, Chen Lv, Juaji Wang, Dongpu Cao, Efstathios Velenis, Fei-Yue Wang	The experimental images are collected using a low-cost camera, and ten drivers are involved in the naturalistic data collection. The raw images are segmented using the Gaussian mixture model to extract the driver body from the background before training the behavior recognition CNN model. To reduce the training cost, transfer learning method is applied to fine tune the pre-trained CNN models. Three different pre-trained CNN models, namely, AlexNet, GoogLeNet, and ResNet50 are adopted and evaluated.
Looking at Hands in autonomous vehicles: A ConvNet approach using part affinity fields.	Kevan Yuen, Mohan Trivedi	The paper introduces a fast ConvNet approach based on OpenPose by Cao, et.al. for full body joint estimation. The network is modified with fewer parameters and retained using their own day-time naturalistic autonomous driving dataset to estimate joint and affinity heatmaps for driver’s wrist and elbow.
Driver distraction detection using semi supervised machine learning	Tianchi Liu, Yan Yang, Guang-Bin Huang, Yong Kiang Yeo, and Zhiping Lin	The images are classified based on eye movement, facial expressions to determine the activity of the driver. Since, the model uses a semi supervised machine learning method, it is efficient in terms of cost and time.

LITERATURE REVIEW

PAPER	AUTHORS	DESCRIPTION
Driver activity recognition using deep learning	]Satoshi Suzuki, Yukie Amemiya, Maiko Sato	Driver's status is detected by recognizing the skin color of the driver, then using transfer learning to match the image with sample data.
Real time system for robust multiple face detection, tracking and hand posture recognition in color video sequence	J-C Terrillon, A.Pilpre, Y.Niwa, K.Yamamoto	The system relies on the three fundamental cues of color, shapes and motion and integrates three mutually complementary sub-systems, in order to achieve high rates of detection, tracking and recognition.
Capturing car-following behaviors by deep learning	X. Wang, R. Jiang, L. Li, Y. Lin, X. Zheng, and F.-Y. Wang	A deep neural network-based car following model is proposed that uses two categories to analyze the situation, a. velocity difference and position difference b. driver body movement.

EXISTING SYSTEMS

Yang Xing, Chen Lv, Juaji Wang, Dongpu Cao, Efstathios Velenis, Fei-Yue Wang, "Driver activity recognition for Intelligent Vehicles: A deep learning approach", IEEE Transactions on vehicular technology, Vol. 68, No. 68, 6 June 2019.

A driver activities recognition system is designed based on the deep convolutional neural networks (CNN) in this paper. Specifically, seven common driving activities are identified, which are the normal driving, right mirror checking, rear mirror checking, left mirror checking, using in-vehicle radio device, texting, and answering the mobile phone, respectively. Among these activities, the first four are regarded as normal driving tasks, while the rest three are classified into the distraction group. The experimental images are collected using a low-cost camera, and ten drivers are involved in the naturalistic data collection. The raw images are segmented using the Gaussian mixture model to extract the driver body from the background before training the behavior recognition CNN model. To reduce the training cost, transfer learning method is applied to fine tune the pre-trained CNN models. Three different pre-trained CNN models, namely, AlexNet, GoogLeNet, and ResNet50 are adopted and evaluated. The detection results for the seven tasks achieved an average of 81.6% accuracy using the AlexNet, 78.6% and 74.9% accuracy using the GoogLeNet and ResNet50, respectively. Then, the CNN models are trained for the binary classification task and identify whether the driver is being distracted or not. The binary detection rate achieved 91.4% accuracy, which shows the advantages of using the proposed deep learning approach. Finally, the real-world application are analyzed and discussed.

PROPOSED SYSTEMS

The model consists of four major stages Data collection for the neural network to learn, training the CNN model to the sample data, using the trained model to generate fuzzy values, image classification based on the fuzzy values. Before building a CNN model we have to import the necessary system, machine learning, mathematical, and graphical representation libraries.

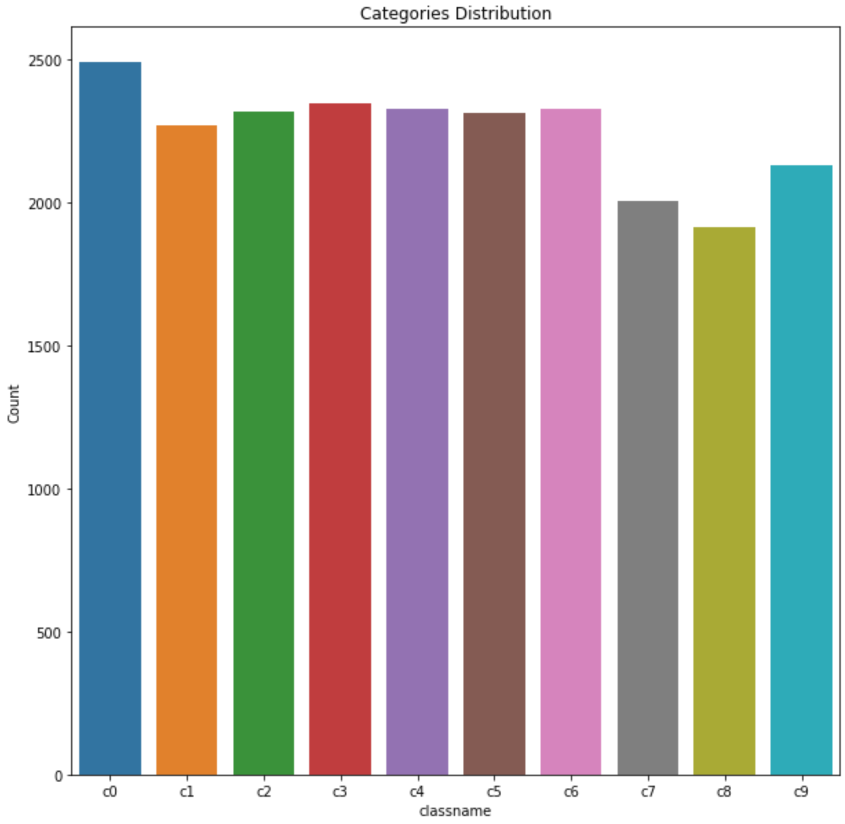
Data collection:

Initially, we are collecting the possible distractions of the drivers while driving, like

- texting-right
- texting-left
- talking on phone-right
- talking on phone-left
- operating radio
- drinking
- reaching behind
- hair adjustment and makeup
- talking to passengers

PROPOSED SYSTEMS (CONTD)

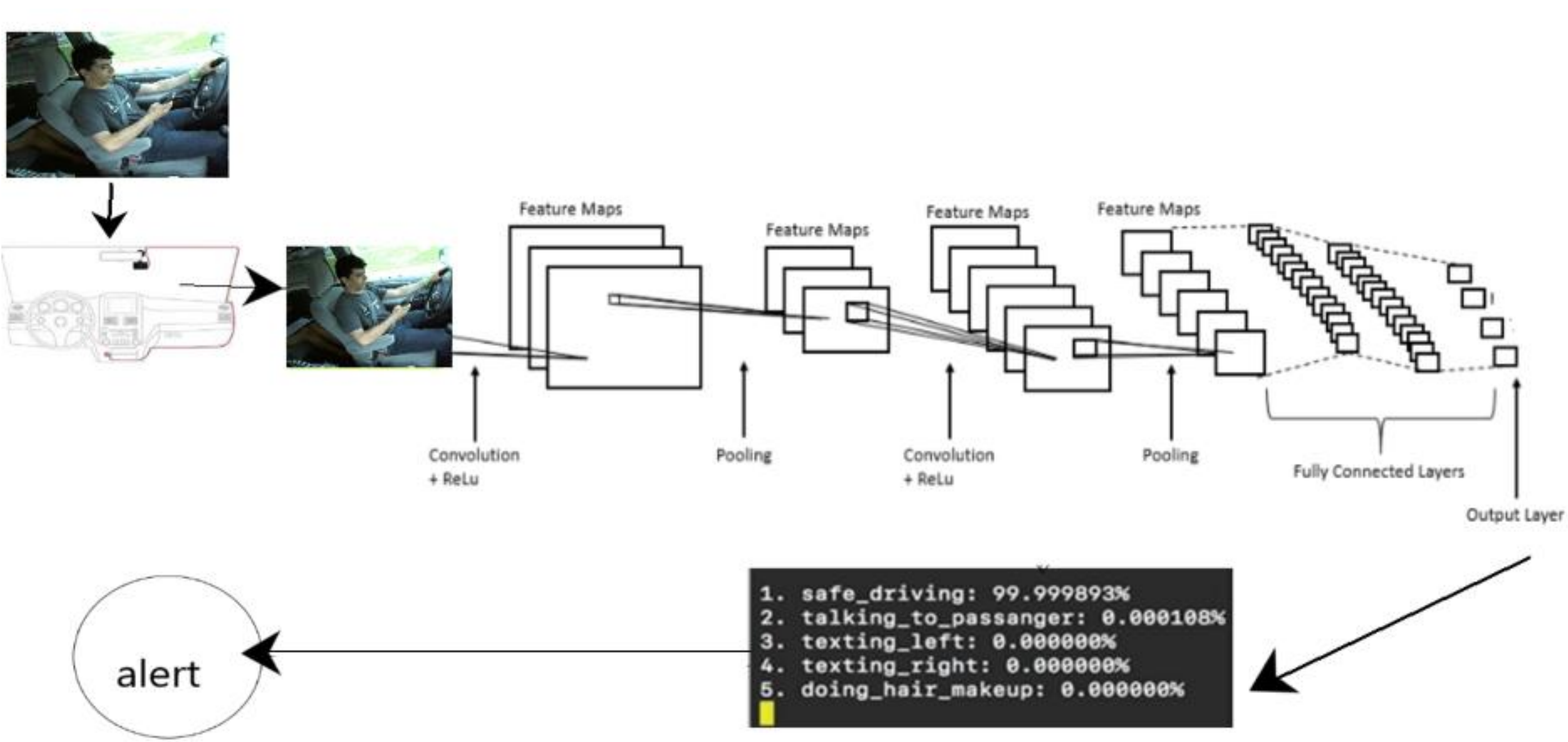
We also collect the images of safe driving drivers. All these images are collected and categorized based on their distraction and stored in a folder. Every distraction belongs to a class name and since, we have nine distractions we have classes *c1*, *c2*, *c3*, *c4*, *c5*, *c6*, *c7*, *c8*, *c9* which are the distracted classes and *c0* class is safe driving that holds the image of safe driving drivers. The statistics of the images collected are shown below.



PROPOSED ARCHITECTURE

With the collected images we are going to build and train a simple CNN to detect whether the driver is distracted or not if he/she is distracted the model will classify it to the desire distracted group. The initial CNN model consists of following layers

Input→ Convolution→ ReLu→ Pooling→ Convolution → ReLu → Pooling→ Convolution→ ReLu→ Pooling→Convolution→ReLu→Pooling→Dropout→Flatten→Fully Connected.



MODULE EXPLANATION

Initially, the images are fed into the input layer which is directed to four series of Convolution layer, ReLu, Pooling layer.

In the convolution layer, the image is converted to 64X62 image matrix and this matrix is subjected to an activation function ReLu and gets converted into 128X128, 256X256, 512X512 image matrices, each image matrix gets into activation function before getting converted into another matrix size. The output from every series of layers is maxpooled and sent as an input to another series of layers. At the end the output is flattened and dropout is done to avoid overfitting and probability of the image belong to each class is known and the best-valued class is displayed.

Here, the prediction of distraction class is wrong this may because of some noise in the images or our approach is not correct, therefore another CNN model is built and trained with the data collected, the newly built CNN model is built focused on gaussian method the effectively reduces the noise in the image and separates human body from background which makes the result more accurate. This model consists of series of layers as follows,

Input→ Convolution layer→ Batch normalization→ Convolution layer→Batch Normalization→ Maxpooling→ Dropout→ Convolution layer→Batch normalization→ Convolution layer→ Batch Normalization→Maxpooling→Dropout→Convolution layer→Batch normalization→Convolution layer→ Batch Normalization→ Maxpooling→ Dropout→ Flatten→ Dense→ Batch Normalization→ Dropout→ Dense→ Dropout→ Dense→ Output.

MODULE EXPLANATION (contd.,)

Here, *Convolution layer*→ *Batch normalization*→ *Convolution layer*→ *Batch Normalization*→ *Maxpooling*→ *Dropout*

makes a layer which is the processing layer, where the image is converted into 2d matrix of 32X32 and sent to activation function ReLu and batch normalization is done to reduce the noise for effective mathematical computation. The image is again fed in to convolution layer as image matrix of same size as above and again batch normalization is made and sent to maxpool layer, where the output from each node in this layer is combined and sent as a input to other subsequent layer, Dropout is done to avoid over fitting of the model.

The same process is again carried out for another two layers with an image matrix of size 64X64 and 128X128.

Image classification based on fuzzy values:

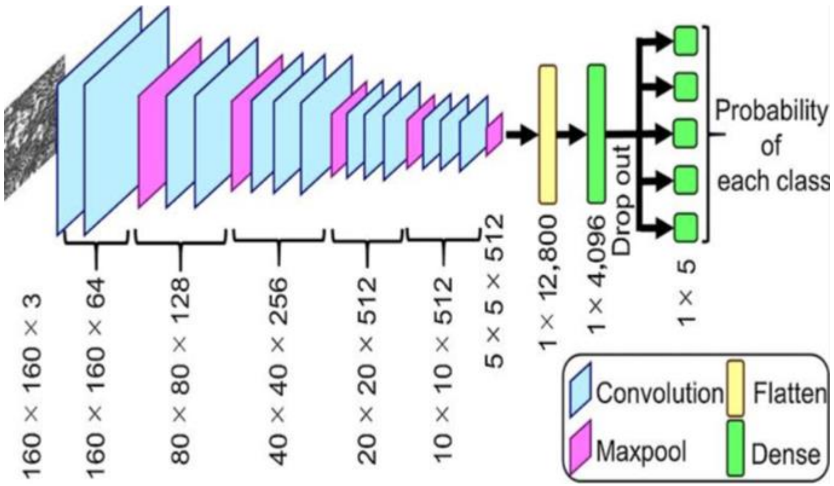
Finally, the output layer consists of,

Flatten→*Dense*→*BatchNormalization*→*Dropout*→ *Dense*→*Dropout*→*Dense*→*Output*

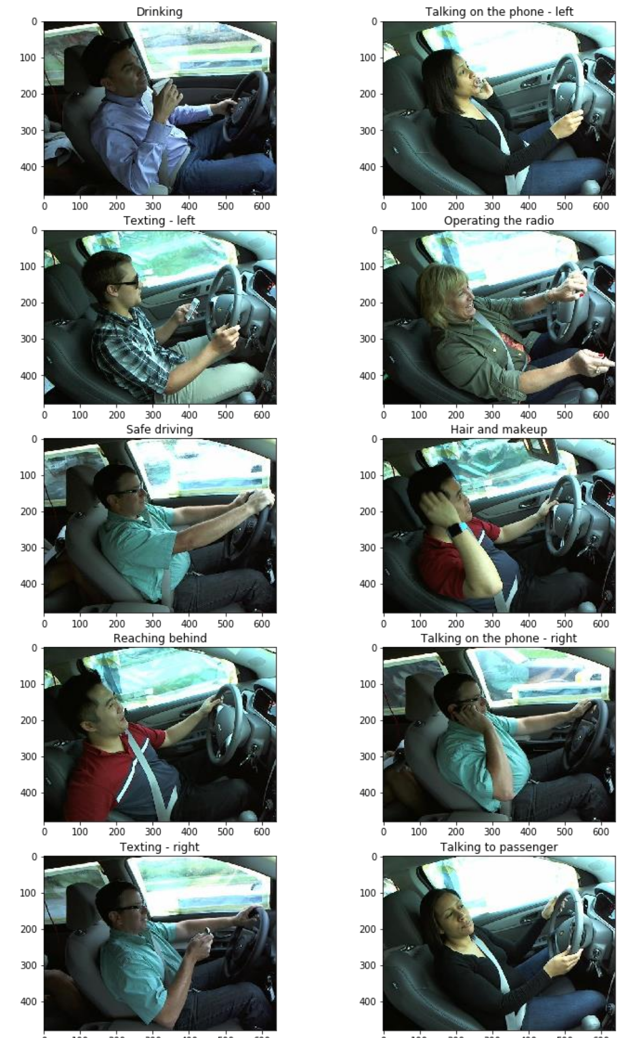
where the image matrix is flattened and densed, which makes them capable to calculate the fuzzy values and deliver the output.

MODULE EXPLANATION

- Layers of the CNN



Distractions



EXPERIMENTS AND RESULTS

Building the First CNN

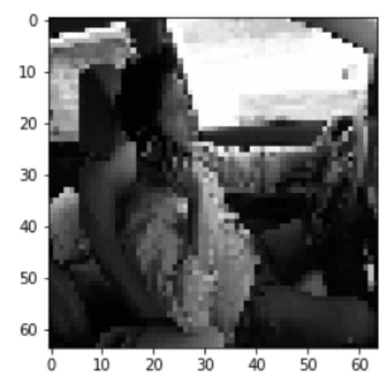
```
Instructions for updating:
Colocations handled automatically by placer.
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 64, 64, 64)        640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 256)       295168
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 8, 8, 256)         0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 512)         1180160
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 4, 4, 512)         0
_________________________________________________________________
dropout_1 (Dropout)          (None, 4, 4, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_1 (Dense)              (None, 500)               4096500
_________________________________________________________________
dropout_2 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                5010
=================================================================
Total params: 5,651,334
Trainable params: 5,651,334
Non-trainable params: 0
```

Training the First CNN

```
Train on 17939 samples, validate on 4485 samples
Epoch 1/10
17939/17939 [==============================] - 525s 29ms/step - loss: 14.4470 - accuracy: 0.1034 - val_loss: 14.4
434 - val_accuracy: 0.1039

Epoch 00001: val_loss improved from inf to 14.44340, saving model to saved_models/weights_best_vanilla.hdf5
Epoch 2/10
17939/17939 [==============================] - 532s 30ms/step - loss: 14.4451 - accuracy: 0.1038 - val_loss: 14.4
434 - val_accuracy: 0.1039

Epoch 00002: val_loss did not improve from 14.44340
Epoch 3/10
17939/17939 [==============================] - 582s 32ms/step - loss: 14.4505 - accuracy: 0.1035 - val_loss: 14.4
434 - val_accuracy: 0.1039

Epoch 00003: val_loss did not improve from 14.44340
Epoch 00003: early stopping
```

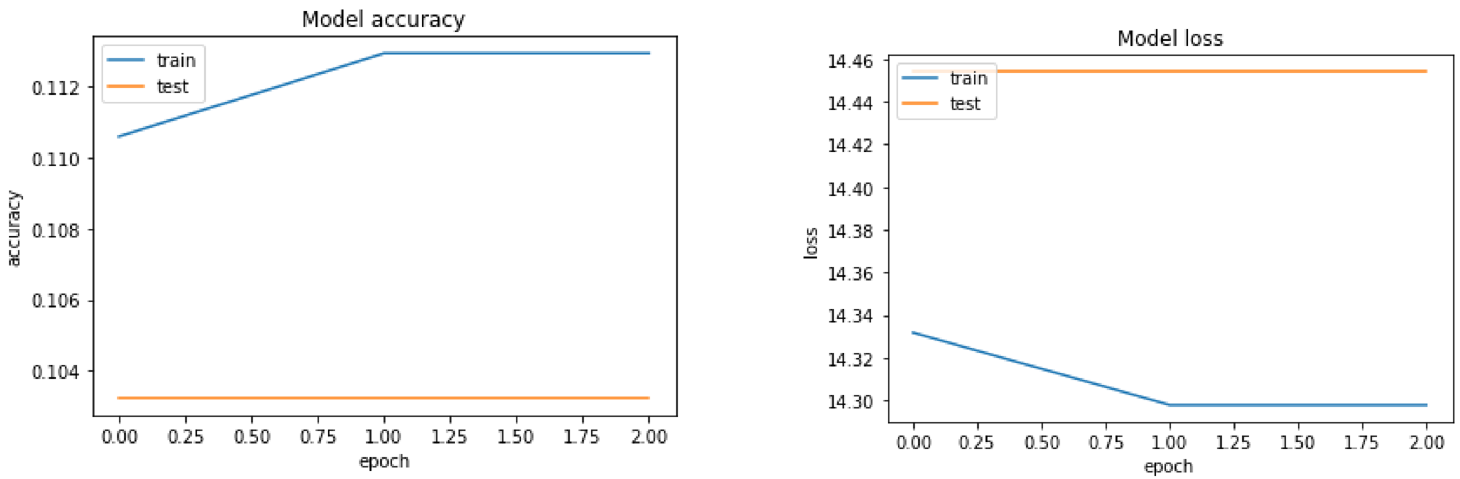
Output of the First CNN

```
1/1 [==============================] - 0s 157ms/step
Y prediction: [[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]]
Predicted: Operating the radio
```



EXPERIMENTS AND RESULTS

Graphs showing model accuracy and model loss



EXPERIMENTS AND RESULTS

Building the Second CNN

```
Train on 17939 samples, validate on 4485 samples
Epoch 1/10
17939/17939 [==============================] - 718s 40ms/step - loss: 1.2675 - accuracy: 0.5893 - val_loss: 0.4496
- val_accuracy: 0.8615

Epoch 00001: val_loss improved from 14.71652 to 0.44964, saving model to saved_models/weights_best_vanilla.hdf5
Epoch 2/10
17939/17939 [==============================] - 712s 40ms/step - loss: 0.3538 - accuracy: 0.8850 - val_loss: 0.2387
- val_accuracy: 0.9342

Epoch 00002: val_loss improved from 0.44964 to 0.23870, saving model to saved_models/weights_best_vanilla.hdf5
Epoch 3/10
17939/17939 [==============================] - 715s 40ms/step - loss: 0.2124 - accuracy: 0.9363 - val_loss: 0.0763
- val_accuracy: 0.9835

Epoch 00003: val_loss improved from 0.23870 to 0.07635, saving model to saved_models/weights_best_vanilla.hdf5
Epoch 4/10
17939/17939 [==============================] - 730s 41ms/step - loss: 0.1565 - accuracy: 0.9523 - val_loss: 0.0654
- val_accuracy: 0.9853

Epoch 00004: val_loss improved from 0.07635 to 0.06544, saving model to saved_models/weights_best_vanilla.hdf5
Epoch 5/10
17939/17939 [==============================] - 887s 49ms/step - loss: 0.1287 - accuracy: 0.9604 - val_loss: 0.0549
- val_accuracy: 0.9875

Epoch 00005: val_loss improved from 0.06544 to 0.05493, saving model to saved_models/weights_best_vanilla.hdf5
Epoch 6/10
17939/17939 [==============================] - 936s 52ms/step - loss: 0.1107 - accuracy: 0.9687 - val_loss: 0.0631
- val_accuracy: 0.9851

Epoch 00006: val_loss did not improve from 0.05493
Epoch 7/10
17939/17939 [==============================] - 763s 43ms/step - loss: 0.0998 - accuracy: 0.9699 - val_loss: 0.0479
- val_accuracy: 0.9884

Epoch 00007: val_loss improved from 0.05493 to 0.04792, saving model to saved_models/weights_best_vanilla.hdf5
Epoch 8/10
17939/17939 [==============================] - 694s 39ms/step - loss: 0.0871 - accuracy: 0.9760 - val_loss: 0.0832
- val_accuracy: 0.9844

Epoch 00008: val_loss did not improve from 0.04792
Epoch 9/10
17939/17939 [==============================] - 666s 37ms/step - loss: 0.0796 - accuracy: 0.9774 - val_loss: 0.0525
- val_accuracy: 0.9900

Epoch 00009: val_loss did not improve from 0.04792
Epoch 00009: early stopping
```

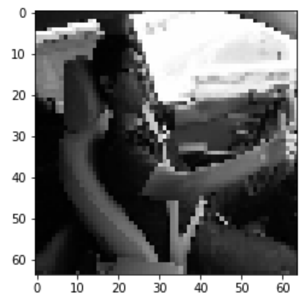
Training the second CNN

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 62, 62, 32)        320
_________________________________________________________________
batch_normalization_1 (Batch (None, 62, 62, 32)        128
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 62, 62, 32)        9248
_________________________________________________________________
batch_normalization_2 (Batch (None, 62, 62, 32)        128
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 31, 31, 32)        0
_________________________________________________________________
dropout_3 (Dropout)          (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 31, 31, 64)        18496
_________________________________________________________________
batch_normalization_3 (Batch (None, 31, 31, 64)        256
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 31, 31, 64)        36928
_________________________________________________________________
batch_normalization_4 (Batch (None, 31, 31, 64)        256
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
dropout_4 (Dropout)          (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 16, 16, 128)       73856
_________________________________________________________________
batch_normalization_5 (Batch (None, 16, 16, 128)       512
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 16, 16, 128)       147584
_________________________________________________________________
batch_normalization_6 (Batch (None, 16, 16, 128)       512
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 8, 8, 128)         0
_________________________________________________________________
dropout_5 (Dropout)          (None, 8, 8, 128)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_3 (Dense)              (None, 512)               4194816
_________________________________________________________________
batch_normalization_7 (Batch (None, 512)               2048
_________________________________________________________________
dropout_6 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_4 (Dense)              (None, 128)               65664
_________________________________________________________________
dropout_7 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_5 (Dense)              (None, 10)                1290
=================================================================
Total params: 4,552,042
Trainable params: 4,550,122
Non-trainable params: 1,920
```

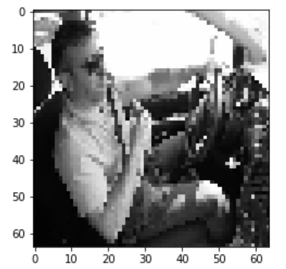
EXPERIMENTS AND RESULTS

Outputs of the Second CNN

```
1/1 [==============================] - 0s 10ms/step
Y prediction: [[3.95868719e-02 1.16372485e-06 1.43047961e-04 4.69134829e-05
  9.57350969e-01 2.95145605e-06 1.34393427e-04 1.16825014e-07
  3.74381983e-04 2.35918094e-03]]
Predicted: Talking on the phone - left
```



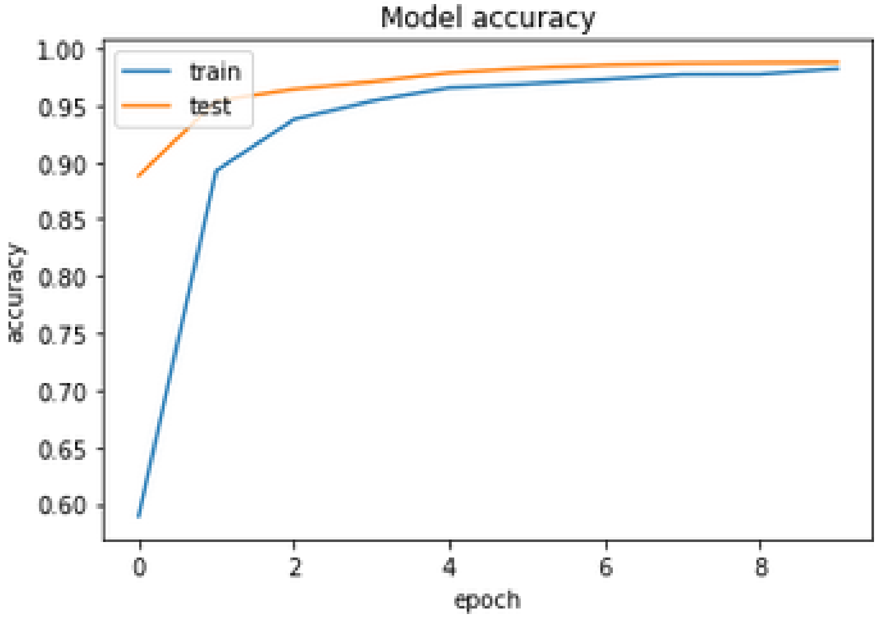
```
1/1 [==============================] - 0s 14ms/step
Y prediction: [[3.1464564e-11 9.3127660e-08 6.2361693e-08 9.5179525e-15 3.8761104e-07
  2.9444767e-12 9.9999452e-01 1.2184723e-11 4.8580541e-06 3.1930356e-09]]
Predicted: Drinking
```

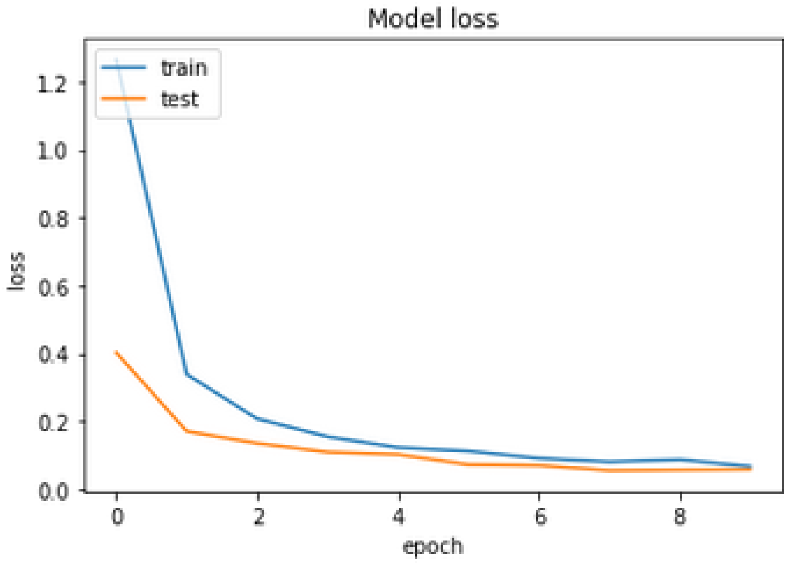


```
1/1 [==============================] - 0s 13ms/step
Y prediction: [[4.2842516e-05 4.4411495e-06 8.7053431e-03 8.6527389e-06 1.7317489e-03
  9.7576058e-01 3.2313255e-04 4.6716340e-07 8.5623446e-04 1.2566611e-02]]
Predicted: Operating the radio
```

EXPERIMENTS AND RESULTS

Grpahs showing model accuracy and model loss





CONCLUSION

- In this work, a driver activity recognition system based on 2-layer deep Convolutional Neural Network model is proposed. The model uses two layers of convolutional neural networks to identify the driver activity based on pre-learned sample data.
- The fuzzy values thus generated are used to determine the activity that the driver is being involved so as to alert the driver or take precautionary measures as a attempt to reduce vehicular accidents.

REFERENCE

- [1] Yang Xing, Chen Lv, Juaji Wang, Dongpu Cao, Efstathios Velenis, Fei-Yue Wang, “Driver activity recognition for Intelligent Vehicles: A deep learning approach”, IEEE Transactions on vehicular technology, Vol. 68, No. 68, 6 June 2019.
- [2] Kevan Yuen, Mohan Trivedi, “Looking at Hands in autonomous vehicles: A ConvNet approach using part affinity fields.”, IEEE transaction on intelligent vehicles, 2019.
- [3]Ahmed Youssef, Jean-marie aerts, Barts Vanrumste, Stijn Luca, “A localized learning approach applied to human activity recognition”, IEEE Intelligent systems, IEEE (Early access), 2020.
- [4] Felix-constantin Adochieei, Ioana Raluca Adochiei, Radu ciucu, Gladiola pietroiu-andruseac, Florin Ciprian Argatu, Nicole Jula, “Design and implementation of a Body posture detection system”, 2019 E-health and bio-engineering conference (EHB), lasi, Romania, 2019, pp.1-4.
- [5] Yukun Zheng, HongXiang Yuan, Rui Song, Xin Ma, Yibin Li, “A comparative study of Hand gestures recognition based on MobileNetV2 and ConvNet Models.”, IEEE International conference on I mage and signal processing and their applications, 2019, pp. 1-6.
- [6] Chao Ma, Jie Yao, Shaohong Wang, “Dynamic analysis on the whole body with sitting posture for impact loading”, The journal of Engineering, no 23, pp.8743-8746, IEEE, 2019.
- [7] Chao Ma, Jie Yao, Shaohong Wang, “Dynamic analysis on the whole body with sitting posture for impact loading.”, The journal of engineering, volume 2019, issue 23, pp. 8743-8746, 12 2019.
- [8] Haojie Li, Qijie zhao, Weichi Zhao, Yijing Wu, “Driver-car Natural interaction method based on Head-eye Behaviors”, IEEE Intelligent Human-Machine Systems and cybernetics, Hangzhou, China, 2019, pp. 161-164.
- [9] Ling Xie, Xiao Guo, “Object detection and analaysis of human body postures based on TensorFlow.”, 2019 IEE international conference on Small Internet of Things (SmartIOT), Tianjin, China, 2019, pp.397-401.
- [10] Satoshi Suzuki, Yukie Amemiya, Maiko Sato, “Enhancement of gross-motor action recognition by CNN with openpose”, 45th Annual conference of the IEEE Industrial Electronics society, Lisbon, Portugul, vol.1, 2019, pp.5382-5387.
- [11] Tianchi Liu, Yan Yang, Guang-Bin Huang, Yong Kiang Yeo, and Zhiping Lin, “Driver Distraction Detection Using Semi-Supervised Machine Learning”, IEEE Transactions on intelligent transportation systems, Vol. 17, no. 4 Oct.2017

REFERENCE

- [12] Whui Kim, Hyun-Kyun Choi, Byung-Tae Jang, Jinsu Lim, “Driver Distraction detection using convolutional neural network”, IEEE International conference on Information and communication technology convergence, 2017.
- [13] F.-Y. Wang, N.-N. Zheng, D. Cao, C. M. Martinez, L. Li, and T. Liu, “Parallel driving in CPSS: A unified approach for transport automation and vehicle intelligence,” IEEE/CAA J. Automatica Sinica, vol. 4, no. 4, pp. 577–587, Oct. 2017.
- [14] J.Wang, J.Wang, R.Wang, and C. Hu, “A framework of vehicle trajectory replanning in lane exchanging with considerations of driver characteristics,” IEEE Trans. Vehicular Technology vol. 66, no. 5, pp. 3583–3596, May 2017.
- [15] A. Koesdwiady, R. Soua, F. Karray, and M. S. Kamel, “Recent trends in driver safety monitoring systems: State-of-the-art and challenges,” IEEE Transaction on Vehicular Technology vol. 66, no. 6, pp. 4550–4563, Jun. 2017.
- [16] X. Wang, R. Jiang, L. Li, Y. Lin, X. Zheng, and F.-Y. Wang, “Capturing car-following behaviors by deep learning,” IEEE Transaction Intelligent Transport Systems vol. 19, no. 3, pp. 910–920, Mar. 2018.
- [17] Lutfun Nahar, Ria palit, Aymun kafil, Zinnia sultana, Shamima akter, “Real time Driver drowsiness monitoring system by eye tracking using mean shift algorithm”, IEEE 5th International conference on advances in electrical engineering, Dhaka, pp 27-31, 2019.
- [18] Rahman, Amna, Mehran sirshar, Aliya khan, “Real time drowsiness detection using eye blinking monitoring”, Software Engneering conference (NSEC), 2015 National, pp 1-7, IEEE 2015.
- [19] Jun-juh Yan, Hang-hong Kuo, Ying-fan Lin, The-lu Liao, “Real tim driver drowsiness detection system based on PERCLOS and grayscale image processing”, International Symposium in computer, Consumer and control (IS3C), Xián, 2016, pp. 243-246.
- [20] J-C Terrillon, A.Pilpre, Y.Niwa, K.Yamamoto, “DRUIDE: a real time system for robust multiple face detection, tracking and hand posture recognition in color video sequence”, Proceedings of the 17th International Conference on Pattern Recognition, Cambridge, pp. 302-305, vol.3.
- [21] Khalil Bousbau, Mostefa Merah, “A Comparitive study of Hand Gestures Recognition based on MobileNetV2 and ConvNet Models”, IEEE 6th International conference on Image and Signal Processing and their Applications, 2019, pp. 1-6.
- [22] Jun-Hyeong Do, Zeungnam bien, “Effective cue Integration for fast and robust face detection in videos.”,Resuse and Integration, Las vegas, IL, 2007, pp.354-359.

THANK YOU!!