DRIVER ACTIVITY RECOGNITION FOR VEHICLES USING DEEP CNN MODELS

[1]R Gnanavel, [2]G Kalpana [3]MR Ravisankar, [4]K Naveen Kumar, [5]NS Ramasamy.

[1,2]Assistant Professors, Department of Computer Science and Engineering, Rajalakshmi Institute of Technology, Chennai, India.

[3,4,5]UG Students, Department of Computer Science and Engineering, Rajalakshmi Institute of Technology, Chennai, India.

[1]rgvelu22@gmail.com [2]g.gkalpana@gmail.com [3]ravisankarmaru@gmail.com [4]naveen55k98@gmail.com [5]vishak12.1999@gmail.com

Abstract – **Most Important factors that affect car passenger safety are Driver decisions and behaviours. Common activities that the drivers are involved in include – Safe driving[1], texting on mobile phones[2], speaking on mobile phone[3], operating the radio[4], drinking or eating[5], reaching behind[6], adjusting makeup or hair[7], or talking to his/her fellow passengers[8]. Among these activities, the first four are regarded as normal driving tasks, while the last seven are classified into the distracted group. The idea of the approach is to receive the live video feed from the dashboard camera, analyze the frames using pre-trained CNN models then fine tune it using transfer learning method, to determine the type of activity that the driver is being involved in. The objective of the project is to observe the driver behaviours, using which a driver activity recognition system was built using deep Convolutional Neural Networks (CNN).**

KEYWORDS: *Image classification, Convolutional neural network, Machine learning, Deep learning.*

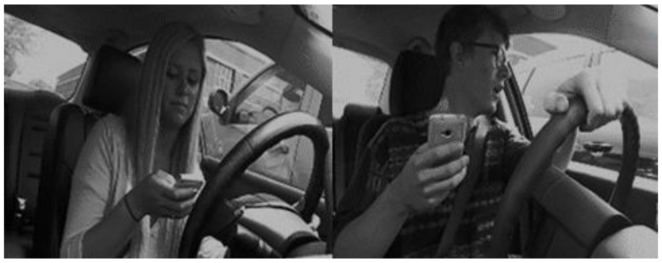


Fig 1: Drivers checking their mobile phones (distraction)

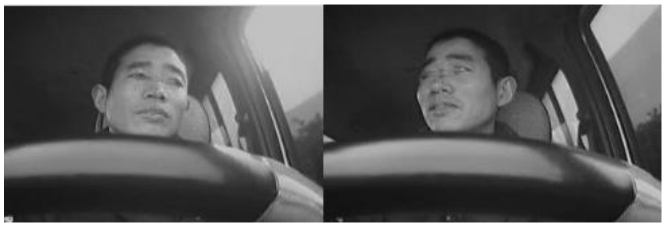


Fig 2: Drivers checking their rear-view mirrors (not a distraction)

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 branched into the three following major types,

Ocular	-	Distraction of sight from road. [1]
Physical	-	Assertion hands from the steering. [2]
Intellectual	-	Lack of concentration while driving. [3]

A low-cost camera is used to collect experimental images, using naturalistic data from drivers. Then, segmentation of raw images is done to remove background and noise from the image and extract the body of the driver alone using the *Gaussian mixture model,* before training the activity recognition *Convolutional Neural Network* model.

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.

RELATED WORKS

Driver distraction detection using single CNN model [1]. The model recognizes the skin color of the driver to detect his/her behavior. Since, it uses a single CNN model, the fuzzy values generated are less accurate.

Shallow learning-based status detection [2]. The model studies the images in cascading stages and uses several classifiers to detect face, eyes, mouth and posture of the driver.

Direct status detection from input image [3]. The method directly detects the driver drowsiness from the input video. This model loses information from the driver's facial landmarks.

Using semi supervised machine learning to detect Driver distractions [4]. 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.

Driver activity recognition using deep learning [5]. Driver's status is detected by recognizing the skin color of the driver, then using transfer learning to match the image with sample data.

Vehicle trajectory re-planning in lane exchange while considering driver characteristics - framework [6]. A system that recognize the driver activities and changes the lane according to the speed and driver alertness.

Using deep learning to Capture car-following behaviors [7]. A car following model using deep neural networks is proposed that uses two categories to analyze the situation, a. velocity difference and position difference b. driver body movement.

Using mean shift algorithm to track eye movement for a driver drowsiness monitoring system [8]. The eye pupil movement are recorded using low cost cameras, then a mean shift algorithm-based clustering technique is used to capture driver drowsiness using eye ball movement.

Real time drowsiness detection using eye blinking monitoring [9]. The eye movement is tracked using low cost black and white camera, and the pattern is matched and clustered with previously recorded data. The compared data re used to detect the state of the driver.

Using grayscale and PERCLOS image processing for Real time driver drowsiness detection [10]. The approximate position of the driver's face in grayscale images are calculated and the eye positions are calculated using small templates to establish a fatigue model, which continuously monitor's the driver's state.

A real time system using DRUIDE on color video sequence for robust multiple face detection, tracking and hand posture recognition [11]. The model uses frame by frame analysis of live feed video fed as input to a neural network The system uses three essential factors such as shapes, color and movement and combine three sub systems that are mutually complimentary, to detect, track and recognize at higher efficiency.

Robust and quicker face detection in videos using effective clue consolidations [12]. A framework that uses multiple clues that consolidate three major factors such as appearance, color and movement which are complimentary features for recognizing the faces. This system is better than other existing systems in reducing false positives, and improving the rate of detection and cost of computation by decreasing the place and scale in which the images are searched.

A localized learning approach applied to human activity recognition [13]. To reduce the influence of imbalance and ambiguity in HAR problems, a novel hybrid localized learning approach of K- nearest neighbors least squares support vector machines is proposed. A hybrid localized learning algorithm is applied to the HAR problem.

Designing and implementing a system that detects body posture [14]. Sensors are used to record real time values, then the values are used to filter and process the data to gather information about driver's posture using calculating algorithms over the time period.

Using MobileNet Version two and Convolutional Networks Models to recognize Hand gestures [15]. In this comparative study, a deep convolutional neural network and Transfer learning is used along with MobileNet that are pre-trained, to recognize limb movement.

Analyzing the Sitting postures dynamically [16]. High speed cameras are used to capture specific parts of the driver's body. The displacement of the recorded parts are taken as parameters and motion, acceleration, speed and velocity of the parts are graphed and then, matched with previously learned graphs.

Head and eye movement recognition using Natural interaction method of the driver towards the car [17]. Infrared Cameras are used to collect Two-Dimensional images of the driver's head and Data of the Three-Dimensional point cloud. Then, SDM minimization of nonlinear least square functions are used to recognize the human face in the captured data. The feature points of the eyes are pupil movement is recorded by using a extensive CV model.

Using TensorFlow to detect and analyze human body postures [18]. TensorFlow is used to develop Deep learning systems that can replace OpenPose. Multiple sets of experimental schemas are built using tuning of algorithms that can classify and detect human postures.

Using OPENPOSE with Convolutional neural networks to recognize gross-motor actions [19]. Recognition of Gross motor action is done using a assessment system based on AI. The system automates the algorithm used for Human body displacement tracking. This vastly improves the efficiency of the model.

A Convolutional Network to analyze affinity fields for autonomous vehicles [20]. This work proposes a fast CNN using OpenPose to joint estimate the movement of whole body. The network is fine tuned using driving datasets for autonomous vehicles by analyzing the heatmaps of the driver's elbow and wrist

Effective face detection in videos using effective clue consolidations [21]. The model uses frame by frame analysis of live feed video fed as input to a neural network, which reduces and analyzes the images and matches the 2D byte array of the cropped image with the pre-trained data.

Using MobileNet Version two and Convolutional Networks Models to recognize Hand motions [22]. Differences in Hand movement, size, position and background are classified using Convolutional neural network and MobileNet. The proposed model works upto an accuracy of 98.9%

PROPOSED MODEL

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

[1] right hand texting
[2] left hand texting
[3] talking on phone using right hand
[4] talking on phone using left hand
[5] eating / drinking
[6] radio operation
[7] reaching behind
[8] hair adjustment and makeup
[9] talking to passengers

and 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.

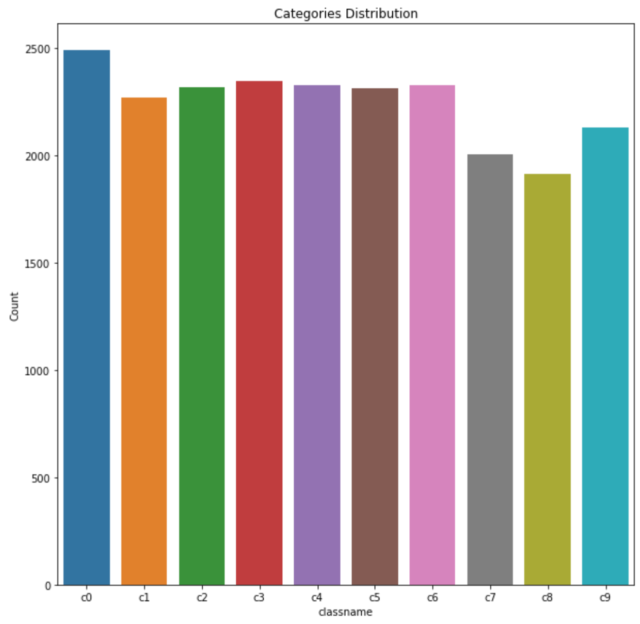


Fig 3. Graph indicating the statistics of the images collected

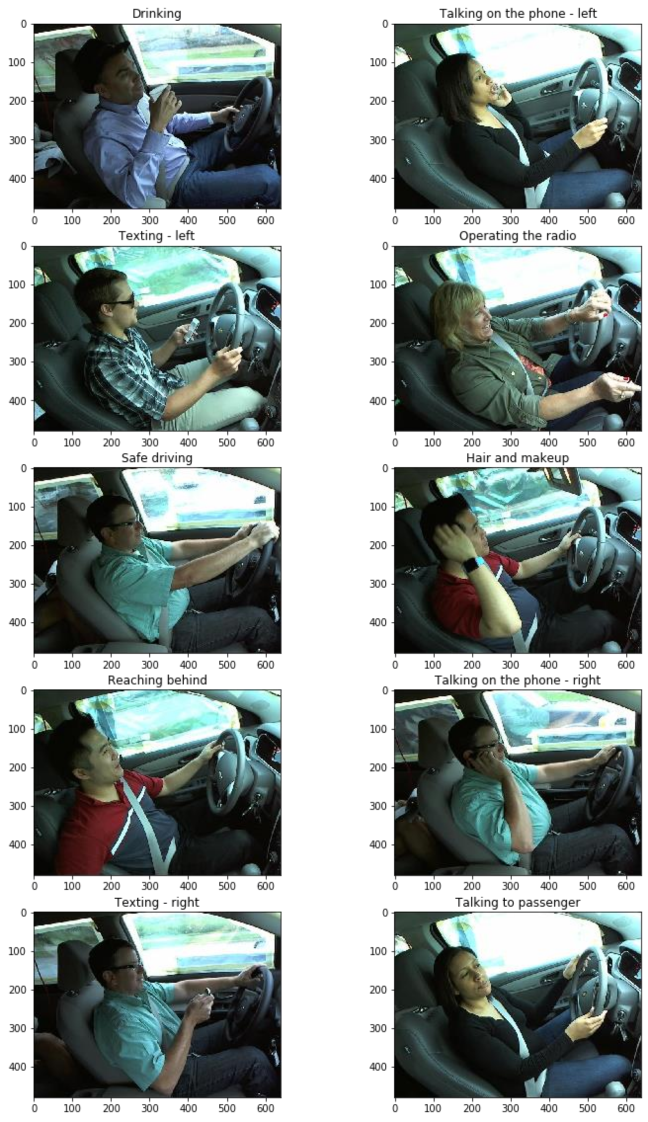
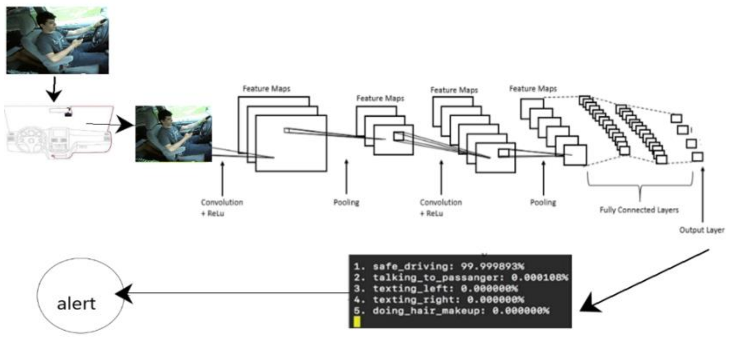


Fig 4. Camera shots of drivers indicating the types of distractions and normal safe driving.

Building and training a CNN model:

With the collected images we are going to build and train a simple CNN to detect the driver's activity. 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.



```
1. safe_driving: 99.999893%
2. talking_to_passanger: 0.000108%
3. texting_left: 0.000000%
4. texting_right: 0.000000%
5. doing_hair_makeup: 0.000000%
```

Fig 5. Architecture diagram

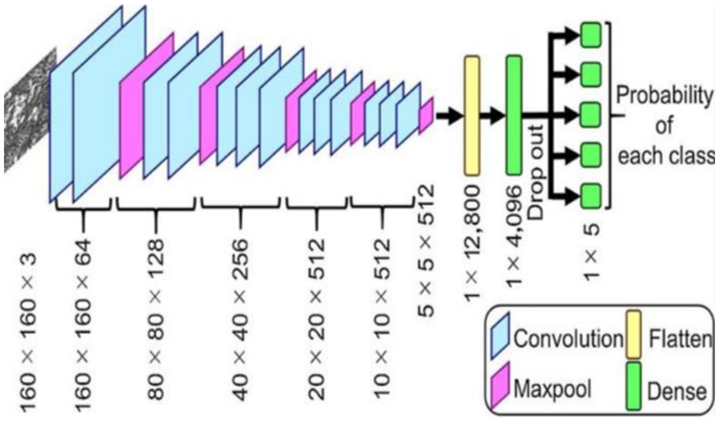
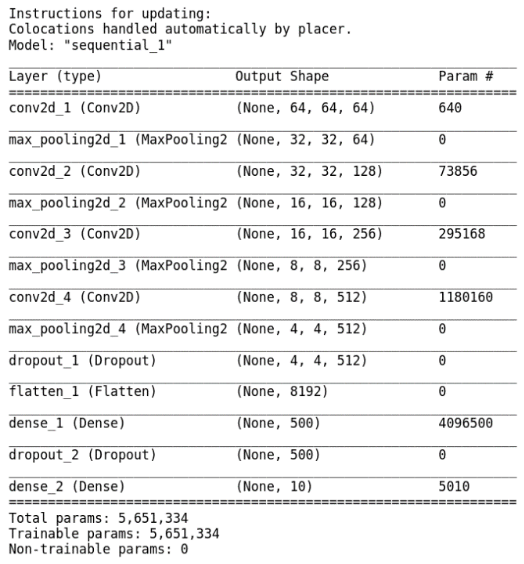


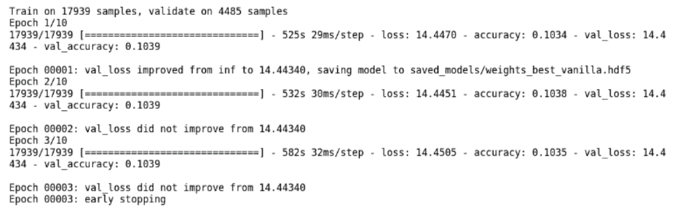
Fig 6. Layers of the Convolutional Neural Network

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.



```
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
```

Fig 7. Building 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
```

Fig 8. Training the first CNN

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

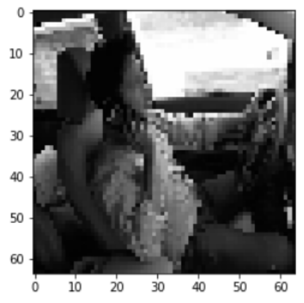


Fig 9. Output produced by the CNN

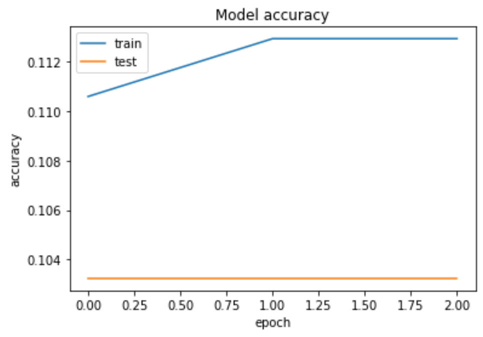


Fig 10. Graph showing Model accuracy

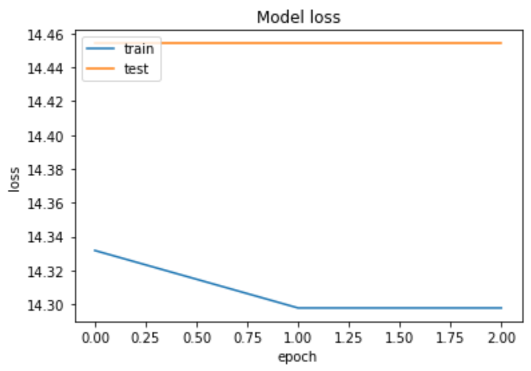


Fig 11. Graph showing Model loss

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

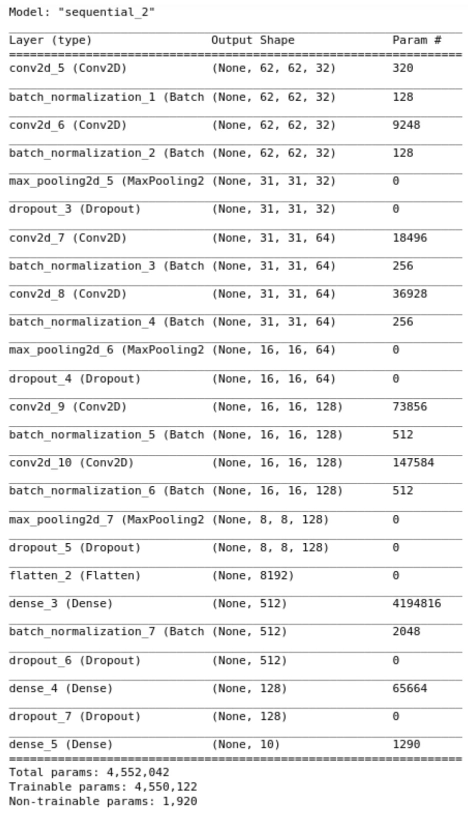
makes a layer which is the processing layer, where the image is converted into 2d matrix of 32X32 and sent to activation function ReLu and batch normalization is done to reduce the noise for effective mathematical computation. The image is again fed in to convolution layer as image matrix of same size 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

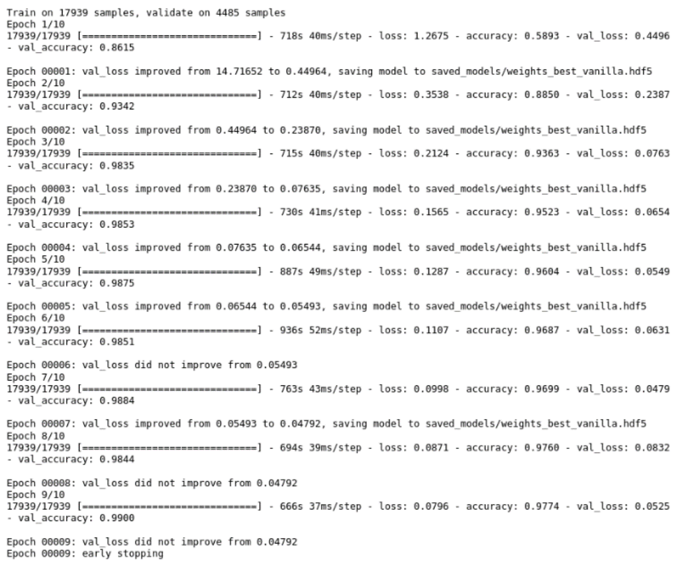
where the image matrix is flattened and densed, which makes them capable to calculate the fuzzy values and deliver the output.



```
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
```

Fig 12. 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
```

Fig 13. Training the second CNN

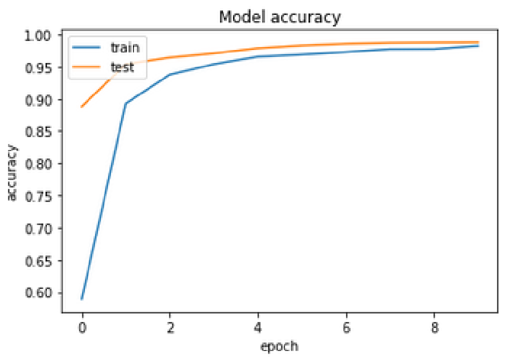


Fig 14. Model accuracy

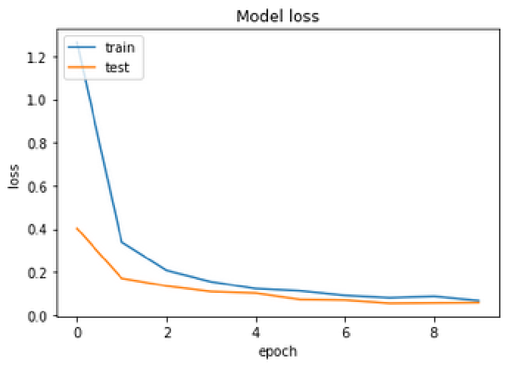


Fig 15. Model loss

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

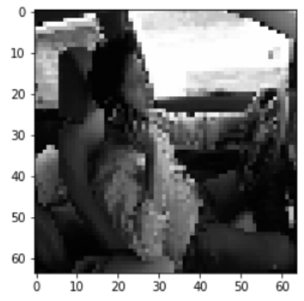


Fig 16. Output produced by the CNN (1)

```
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
```

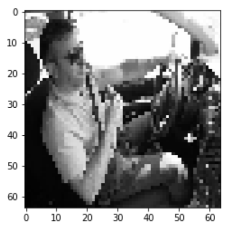


Fig 17. Output produced by the CNN (2)

```
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
```

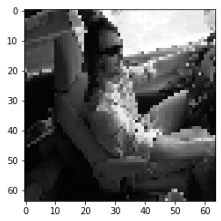


Fig 18. Output produced by the CNN (3)

Therefore, we get a satisfying result from the above model, out of three test images two are correct.

CONCLUSION

In the above proposed model, the driver activity recognition system is based on a multi-layered deep convolution neural network model is proposed. The model uses two layered 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

[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.