Driver Activity Monitoring Using MobileNets

Deval Srivastava, Priyank Shah, and Saim Shaikh

1 Introduction

In our current world as driving technology continues to grow the driving effort required decreases. Hence, drivers become more and more careless resulting in loss of life in many circumstances. The proposed system aims to solve this problem by developing a system to monitor the driver's activity and warn them whenever necessary. The method involves deploying a neural network trained on various categories such as talking or texting on the phone, talking to co-passengers, operating the radio, and drinking water. In our paper, we have extensively tested the performance of different neural networks [1] such as Resnet-50 [2], Inception [3], and MobileNets [4]. Throughout the development, our focus has been to make a system that replicates the driver's real-life conditions. Hence, our network will receive the images from an IR camera allowing our system to perform during nighttime. Our system can be easily fitted to any existing vehicle very easily and will be intuitive to use. Our system has been developed such that it can work even in regions having extremely poor internet connectivity. The system will also be equipped with sensors to detect rash driving and will consist of security features such as fencing, fingerprint authentication to prevent thieving of the vehicle.

From the previous work and research done in this domain, it can be concluded that the most popular computer vision methods include detecting driver inattention using head pose, eye gaze estimation or simply checking eye closure rate as well as measures such as EEG, electrocardiogram, etc. We will discuss these methods and other techniques that have been used in the next section.

The paper has been organized in the following manner next we will be looking at some of the most comprehensive research that has been done in this field and is relevant to this application, Further that the algorithm and its peculiarities will be

D. Srivastava (✉) · P. Shah · S. Shaikh
Fr. Conceicao Rodrigues College of Engineering, Mumbai, India

N. R. Shetty et al. (eds.), *Emerging Research in Computing, Information, Communication and Applications*, Lecture Notes in Electrical Engineering 789,
https://doi.org/10.1007/978-981-16-1338-8_5

explained, after that we can take a look at the complete flow of the system and our design methodology. Post that the dataset, training, and parameters will be discussed. Towards the end, we can look at the results received by us and the conclusion.

2 Literature Survey

According to the survey conducted by us, it can be concluded that the most popular methods to solve this problem involve either driver biological measures, driver physical measures, driving performance measures, or some kind of a hybrid measure [5].

Driver biological measures include biological signals like EEG, electrocardiogram (ECG), electro-oculography (EOG). These signals are collected through electrodes in contact with the skin and then analyzed for fatigue and drowsiness. Physical measures involve eye closure detection and blink frequency, face position, driver gaze to detect inattention.

Driver performance measures involve various measures such as steering angle and other driving criteria. Most research that has been done in the related field has been focused on detecting driver inattention using eye gaze tracking and head pose estimation. These methods rely only on the head and eye movement to detect inattention whereas in real life a driver can be distracted doing various tasks that cannot be detected by head movement alone. It has been observed that current driver monitoring systems employ statistical machine learning methods to detect driver distractions and work on a limited dataset. Research done by Martin et al. [6] involves classifying drivers' gaze into various regions using a machine vision algorithm that utilizes face detection and facial landmark extraction to estimate the gaze of the driver further they are using that to explore driver's gaze dynamic patterns. Further, the authors have condensed gaze dynamics into glance frequencies and duration.

Some research has been done on applying deep learning technologies to solve this problem but such systems cannot be cost-effectively deployed in a vehicle nor do they work in nighttime conditions. These methods have used architectures like the vgg-16 and have been trained on datasets like the StateFarm dataset [7].

3 Algorithm

For the development of our system, we made use of the MobileNet Algorithm [8]. MobileNet [8] is a neural network that was developed by Google to perform on low powered devices lacking graphical GPUs that are known to accelerate neural network performance. MobileNets are small, low-latency, low-power models parameterized to meet the resource constraints of a variety of use cases, one of those use cases is that it can also be deployed on a Raspberry pi which we intend to do.

A standard convolution, filters and combines inputs into a new set of outputs in one step, but in the case of MobileNets it first uses depthwise convolution [8] that applies a single filter to each input channel. The pointwise convolution then applies a 1×1 convolution to combine the outputs of the depthwise convolution. The depth wise separable convolution splits this into two layers, a separate layer for filtering and a separate layer for combining. This factorization has the effect of drastically reducing computation and model size, this modification allows MobileNet to be faster than its other counterparts.

For our application we have employed a MobileNet v2 [8], it's the second iteration of MobileNets and now along with the depthwise separable blocks it also uses bottleneck residual layers and also adds a 1×1 expansion block whose purpose is to expand the number of channels in the data before it goes to the next block. In the proposed system we have used MobileNet v2 as it's much better than its older version, In our mobilenet model we have 17 bottleneck layers which are followed and preceded by convolutional 2d layers and a total of 3.4 M parameters which to put in perspective are far less than 138 M parameters of VGG-16 and 4.2 M parameters of the competing mobilenetV1. The model was on a self-made dataset of drivers performing distracting, reckless activities.

4 Proposed Method

4.1 Implementation

For lucid understanding of the implementation we have developed, the paper presents it in points listed below.

1. Figure 1 describes the entire workflow of the project right from the hardware setup to the user interface. The Pi camera is mounted on an appropriate position in the dashboard of the vehicle. It is then connected to the camera port on the Raspberry Pi. The SM808 GSM + GPS module is connected to the Raspberry Pi via USB TO RS232 serial port. The GPS antenna is connected to the module and placed outside the vehicle with a clear view of the sky. The Raspberry Pi is then connected to a portable power supply via the micro USB port.
2. The Pi camera records footage of the driver and sends the frames to the Preprocessing Unit. The footage is recorded at a resolution of 640*480 at 24 frames per second. The preprocessing unit then performs basic image processing, noise reduction on every third frame and resizes them to 224*224*3 then it is forwarded to the neural network which predicts the class of that image and depending on that result we declare the driver as distracted or not. If the driver is distracted the buzzer is rung to alert the driver.
3. As similar performance is targeted for both night and day time footage. In low light, the Pi Camera is aided by 2 IR bulbs which help provide clear frames even in pitch black conditions. The frames received from the Pi Camera are

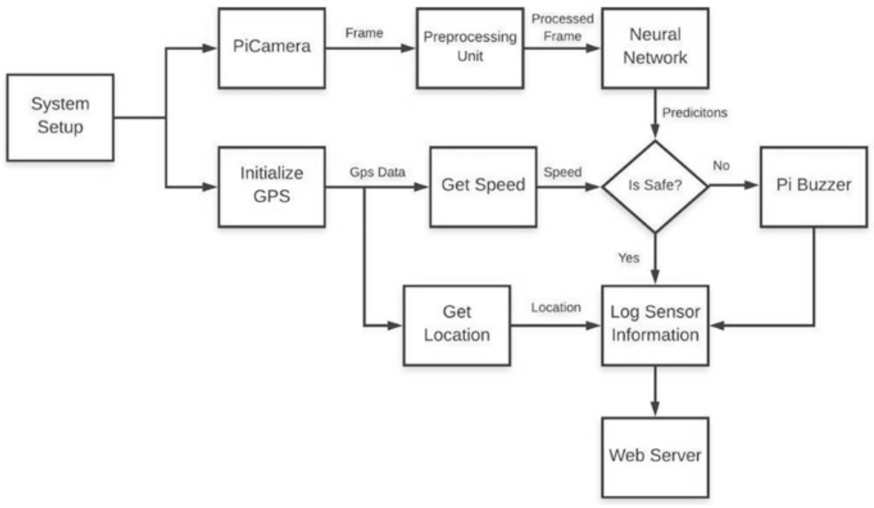

Fig. 1 Implementation flowchart

processed. Alongside this, we also calculate the speed of the vehicle and find out the location of the vehicle the speed is checked to be within the prescribed speed limit and the location within bounds, if either of these details is found to be dissatisfactory then they are reported to the administration with an alert. Once both operations are complete data is pushed to the administrative web server.

4.2 Dataset

When we were looking for a dataset that would be able to suffice our needs for the classification tasks we came across many publicly available datasets one of them being the NTHU Driver Drowsiness Detection Dataset [9]; this dataset consisted of different subjects performing a variety of tasks both during driving many of which can be considered as distracted in our system however one shortcoming that we landed upon while we worked in this dataset was the fact that the dataset was not recorded in an actual car and that factor would affect the performance of the system in real-world cars and secondly this dataset lacked any data points for certain classes we held valuable and were crucial to our research these were the drinking class and use of a smartphone. Overuse of smartphones and drinking while driving are both issues relevant to our current day society. The second dataset that we found was the Kaggle Statefarm Dataset [10]; This dataset included photos of various subjects performing activities in a vehicle where some of them could be considered as reckless and distracting however one of the flaws of this dataset was the lack of any infrared imagery for nighttime and just very poor positioning of the camera

Table 1 Database statistics

S. No.	Class	Count (day + night)
1	Safe driving	5000
2	Talking on phone	5000
3	Texting on phone	5000
4	Drinking	5000
5	Sleeping	5000
6	Yawning	5000

system which would be virtually impossible in a real-world car or truck. While evaluating publicly available datasets we deliberated over each of our options and even considered combining the datasets but none of these approaches proved to be satisfactory as our neural networks trained on these datasets were not performing as expected in real-world conditions [11]. After this realization, we started working on creating a real-world dataset curated to our task. We recorded 6 different drivers performing various distracted activities across multiple cars. We recorded drivers performing activities such as talking on the phone, texting while driving, drinking, sleeping, Yawning. All of these activities were performed in simulated environments where the drivers are not driving. The Dataset was recorded using a pi camera as it's the camera that will be feeding images to the neural network. We recorded images for the night dataset by using an IR camera and IR lights. The database statistics are presented in Table 1 (Figs. 2 and 3).

In the above pictures, we can see the dataset samples from the night and the day. The images are in order of Sleeping, Talking on the phone, Drinking, Texting, Yawning.

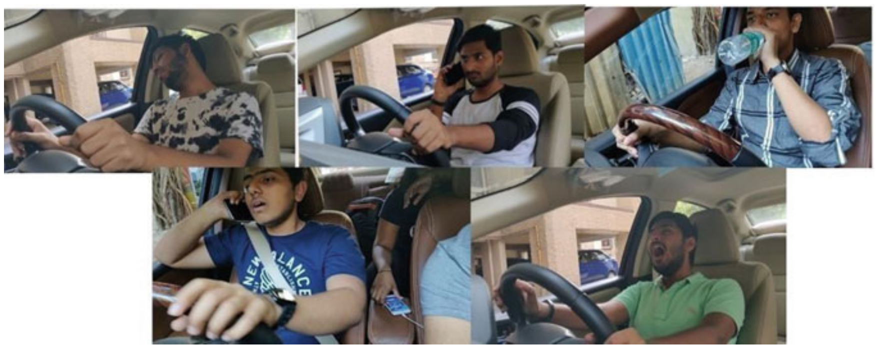

Fig. 2 Dataset samples that were taken during the day

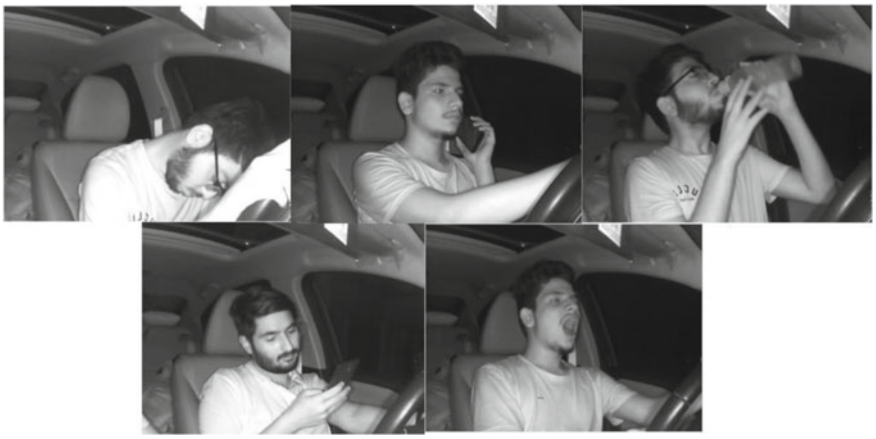

Fig. 3 Dataset samples that were taken during the night

4.3 *Training and Configuration*

This section will discuss how we trained our model on the above dataset. Now coming to train our model we decided to use a transfer learning approach this method says that first, the model has trained on a huge dataset such as the imagenet [12] dataset for image classification tasks, it contains 1.4 M images and 1000 classes allow the model to learn features of images which becomes our base knowledge. The model which has learned these image features now can be carried over and used for other computer vision tasks, this is done precisely by capturing the weights of all the layers but the bottom few fully connected layers, the top layer weights are said to be 'frozen' ie when the model has trained on the new dataset or in this case our dataset the weights will not change and the backpropagation and weight changes will be limited to the bottom layers. Transfer learning allows neural networks to train better and quicker while promoting reusability and modularization.

In our case, we employed a mobilenet V2 model that had been trained on the imagenet dataset further to utilize this model in our application we choose the freeze the bottleneck layers of the model as they are more generalizable and add a layer to average the weights and then a fully connected layer with the softmax activation that will give us the logits.

We have used transfer learning to train our neural network as we have used the model of MobileNet v2 for feature extraction and only trained the last few layers to get the best results and quicker training times. We used 60%, 20%, 20% split for training, validation, and testing respectively for our model. To further simulate real-world conditions we added data augmentation to our model. This allows our model to perform better in difficult scenarios like low lighting, improper camera alignment, etc. After experimenting and testing with a lot of different kinds of augmentations we found the following augmentations gave us the best results were random zoom that generates extra images that are zoomed in randomly up to 20%. In the same way, we added random crops up to 20% and also random brightness for varied conditions. Augmentations effectively increase the size of our dataset and also makes sure the model works better in unknown conditions.

We trained the MobileNet v2 model on our custom dataset with a batch size of 32 with Adam optimization [13] which is known to perform well on similar tasks with a parameter of 0.01, we trained the model on a computer with an Nvidia GTX1070TI with Cuda acceleration. Our model was trained for roughly 6500 steps with a batch size of 32 which took 4 h to train and we stopped when we had a validation accuracy of 95.6%.

5 Results and Discussion

In this section, we will discuss the results we have received after implementing our application to completion. Let's first understand how results are organized, In Table 1 we are looking at all the models we have trained on our datasets to measure up performance and other factors, we have recorded the accuracy of the model on our test split of the dataset, the average CPU usage recorded over 6 h of the utilization of our system and the average frames per second that we were able to process on our system. Further, we also trained our model on various other architectures such as Inception v2, ResNet-50, and VGG-16 [14] model on the same dataset to compare the performance we receive on the Raspberry pi and give a comprehensive result. In a cursory analysis of the table, we can observe that mobilenetV2 offers the highest frames per second and the lowest CPU utilization to go along with it, looking at fps and accuracy metrics of other models we can infer that there is accuracy vs performance tradeoff present here. Higher parameter count and more computationally heavy models such as inception offer higher accuracy to go with it. Coming to our use case there were potential concerns and requirements we had that helped us navigate around the tradeoff. We are deploying the model on a raspberry pi, which is a rather small and weak computer, and along with processing driver images through a neural network, we were also doing auxiliary processing which includes velocity and GPS information calculation, sending regular updates to an administration webserver. With this much processing to go around it becomes crucial to manage CPU resources which are already scarce in raspberry pi. Next concern was regarding the speed of processing frames, in situations where the driver may fall asleep on the wheel we need to be able to act fast and quickly process the data so the driver can be alerted and an accident can be averted, with this in mind it becomes paramount to have high frame per second as to decrease the time spent on other frames before reaching the one deciding frame. Now with consideration to these concerns, we can see that MobileNet v2 is the most ideal neural net model for our use case. We believe that MobileNet v2 was the best model for our application as we want to have the system to be almost real-time which will enable it to prevent accidents. As MobileNet v2 requires the least amount of CPU usage some processing ability of the limited compute on a Raspberry pi CPU can also be used

Table 2 Comparison of different models

S. No.	Model name	CPU usage (%)	Fps	Execution time (min)	Accuracy (%)
1	Inception v2	90	1	8	98
2	VGG-16	93	1	8	96
3	Resnet-50	91	2	4	97
4	**MobileNet v2**	**66**	**6**	**1.34**	**93**

for other tasks such as calculating speed and sending data to a web server. We further tested our system in real-world conditions and found satisfactory results (Table 2).

In the execution period column, we have provided the time it took to run a 60 s video which is encoded with 24 fps and we process every one in three frames. In the second section, we can look at and understand the results we have received in our real-world testing. Firstly, the metric we have developed to compare and analyze results from ours and other competitive solutions in an equitable manner is explained; we recorded over 100 instances of distracted driving doing various activities in both day and nighttime conditions. Next, we ran existing solutions developed for this problem to compare results with our solution. We have chosen solutions utilizing deep learning, eye closure estimation and gaze estimation from head pose for comparison. The deep learning method utilizes a VGG based architecture and has been trained on the NTHU dataset. Since statistical methods involving eye/head estimation cannot generally provide you with the exact distracted activity but rather binary information about whether the driver is focused or not nevertheless we will be using that only for our comparisons. To have fair results we will be considering classes from the existing deep learning solution which are also present in our solution. We believe the sleeping/sleepiness category is an essential element that warrants it to have a separate category.

In Table 3 we can look at the results of each method on detecting other distracting activities and sleeping/sleepiness, we are comparing percentage-based results to assess the number of times each method detected the activity correctly. In a cursory glance, we can infer from this table that in sleeping/sleepiness category are method edges out ahead and moves ahead in the other category with ease, looking at the

Table 3 Comparison with existing solutions

S. No.	Method name	Sleeping/sleepiness (%)	Other distracting activities (%)
1	Deep learning-based method	90	85
2	Eye closure estimation method	88	68
3	Gaze estimation from head pose method	92	72
4	**Our method**	**95**	**93**

underlined entries we can understand from that a deep learning solution performs better on detecting activities but falls short of head pose estimation methods on detecting drowsiness which does well on that task, however, our deep learning solution trained on a custom dataset successfully manages to outperform existing deep learning solutions on activity detection and edges out ahead in drowsiness detection compared to head pose estimation.

6 Conclusion and Future Scope

On successful implementation the system will provide a robust and efficient method to monitor driver activities and thus prevent accidents that occur due to distracted driving, overuse of mobile phones while driving, texting on the phone, drowsiness, sleeping, etc. When such a system is in place it will enforce the drivers to be more careful and drive responsibly which will prevent loss of lives and will promote a safer driving experience for other drivers on the road. Further we can conclude from the results that implementation of mobilenet for this task allows our system to run more efficiently and much quicker compared to contemporary neural network architectures. The decision to go ahead with a custom dataset over open source readily available datasets allows us to add curated classes and more variability and real world features that proved beneficial.

Once applied over a large number of vehicles the system can also be used to create a network of vehicles to share important information. In the future, this network will be able to collect huge amounts of data and this data can be used to plan routes better. Moreover, since we have deployed a hardware platform more and more features can be added in due time. Features such as facial recognition for authentication and various kinds of analysis can be done using the data of our platform. The algorithm in the proposed system relies on a neural network to detect driver's activity which performs well but an object detection approach can be used to detect specific distracting objects which will theoretically perform even better than standard neural net approaches.

References

1. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient based learning applied to document recognition. Proc IEEE 86(11)
2. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR), Las Vegas, NV, pp 770–778. https://doi.org/10.1109/CVPR.2016.90
3. Szegedy C et al (2015) Going deeper with convolutions. In: 2015 IEEE conference on computer vision and pattern recognition (CVPR), Boston, MA, pp 1–9
4. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H (2017). MobileNets: efficient convolutional neural net

5. Ahir A, Gohokar V (2019) Driver inattention monitoring system: a review. In: 2019 international conference on innovative trends and advances in engineering and technology (ICITAET), Shegaon, India, pp 188–194. https://doi.org/10.1109/ICITAET47105.2019.9170249
6. Martin S, Vora S, Yuen K, Trivedi MM (2018) Dynamics of Driver's gaze: explorations in behavior modeling and maneuver prediction. IEEE Trans Intell Veh 3(2):141–150. https://doi.org/10.1109/TIV.2018.2804160
7. Vicente F, Huang Z, Xiong X, Torre F, Zhang W, Levi D (2015) Driver gaze tracking and eyes off the road detection system. IEEE Trans Intell Transp Syst 16(4)
8. Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L (2018) MobileNetV2: inverted residuals and linear bottlenecks. In: 2018 IEEE/CVF conference on computer vision and pattern recognition, Salt Lake City, UT, pp 4510–4520
9. Weng C-H, Lai Y-H, Lai S-H (2016) Driver drowsiness detection via a hierarchical temporal deep belief network. In: Asian conference on computer vision workshop on driver drowsiness detection from video, Taipei, Taiwan, November 2016
10. Kaggle statefarm dataset. https://www.kaggle.com/c/state-farm-distracted-driver-detection
11. Valeriano LC, Napoletano P, Schettini R (2018) Recognition of driver distractions using deep learning. In: 2018 IEEE 8th international conference on consumer electronics—Berlin (ICCE-Berlin), Berlin, pp 1–6. https://doi.org/10.1109/ICCE-Berlin.2018.8576183
12. Deng J, Dong W, Socher R, Li L, Li K, Fei-Fei L (2009) ImageNet: a large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition, Miami, FL, pp 248–255. https://doi.org/10.1109/CVPR.2009.5206848
13. Kingma DP, Ba J (2015) Adam: a method for stochastic optimization. In: ICLR
14. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: International conference on learning representations