Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

Distracted driver detection by combining in-vehicle and image data using deep learning

Furkan Omerustaoglu, C. Okan Sakar [*], Gorkem Kar

Bahcesehir University, Department of Computer Engineering, 34353, Istanbul, Turkey

A R T I C L E I N F O

Article history:
Received 25 December 2019
Received in revised form 9 August 2020
Accepted 15 August 2020
Available online 21 August 2020

Keywords:
Transfer learning
Fine-tuning
Convolutional neural networks
Long short-term memory
Multimodal classification

A B S T R A C T

Distracted driving is among the most important reasons for traffic accidents today. Recently, there is an increasing interest in building driver assistance systems that detect the actions of the drivers and help them drive safer. In these studies, although some distinct data types such as the physical conditions of the driver, audio and visual features, car information are used; the main data source is the images of the driver that include the face, arms, and hands taken with a camera placed inside the car. In this work, we propose to integrate sensor data into the vision-based distracted driver detection model to improve the generalization ability of the system. With this purpose, we created a new data set that includes driver images and sensor data collected from real-world drives. Then, we constructed a two-stage distracted driving detection system to detect nine distracted behaviors. In the first stage, vision-based Convolutional Neural Network (CNN) models were created by transfer learning and fine-tuning methods. In the second stage, Long-Short Term Memory-Recurrent Neural Network (LSTM-RNN) models were created using sensor and image data together. We evaluate our system by two different fusion techniques and show that integrating sensor data to image-based driver detection significantly increases the overall performance with both of the fusion techniques. We also show that the accuracy of the vision-based model increases by fine-tuning the pre-trained CNN model using a related public dataset.

1. Introduction

Distracted driving is, for years, one of the leading causes of car accidents [1]. Some actions carried out by drivers while driving such as using mobile devices, switching to a favorite radio station, eating or drinking may danger road safety and could cause accidents since they require driver's attention. Therefore, detection of these actions might prevent many car accidents.

Due to the severity of the problem, distracted driving detection has received a lot of attention from the research community. In addition to assisting in preventing car accidents, in [2] it is mentioned that driving monitoring and assistance systems help to eliminate distracted driving and thus reduce fuel consumption. Similarly, in [3,4], it is indicated that the driving style of the driver affects fuel consumption to a great extent. In another study, Birrell and Young [5] mentioned that correcting the driving style results in the elimination of distracted driving movements and a reduction in fuel consumption.

Cell phone usage, talking to passengers, texting while driving and operating the radio are among the most common distracted driving behaviors [6,7]. There have been many works to detect such usage by locating the phone using acoustic or cell phone sensing techniques. With eye gaze tracking, the researchers also tried to detect such behavior [8,9]. In most cases, a camera that is shooting the driver while driving is placed inside the car and the taken images are used to classify the driver's actions.

Another approach is to use the sensors embedded in a vehicle. Some cars have hundreds of physical sensors on-board that describes the operation of the internal subsystems of the car. Some of the example sensor readings are speed, revolutions per minute, throttle position, and fuel capacity. These readings can be accessed using the standard On-Board Diagnostics (OBD-II) port available on all vehicles. In related works [10,11], the researchers used these sensors to detect distracting actions. Identifying the driver using the vehicle sensors has also been proposed by some recent works [12–14]. In these works, the authors focused on shared vehicles in professional and commercial settings, and using supervised and unsupervised learning techniques, they identified drivers with an accuracy greater than %90 for up-to 16 driver settings.

In some other recent works [15,16], the authors aimed to classify the driving states as normal or fatigued driving. In [15],

[*] Corresponding author.
E-mail addresses: furkan.omerustaoglu@bahcesehir.edu.tr (F. Omerustaoglu), okan.sakar@eng.bau.edu.tr (C.O. Sakar), gorkem.kar@eng.bau.edu.tr (G. Kar).
URL: https://akademik.bahcesehir.edu.tr/~cosakar/ (C.O. Sakar).

https://doi.org/10.1016/j.asoc.2020.106657
1568-4946/© 2020 Elsevier B.V. All rights reserved.

infrared cameras are used to extract eye region for 26 subjects. By using a CNN with LSTM-RNN units, they could achieve over 95% accuracy in the binary classification task.

The primary goal of our study is to combine vision and sensor-based approaches to improve the distracted driver detection accuracy. For this purpose, we propose a system comprising two key steps. In the first step, we process vision and sensor data separately to build individual detection models. For an efficient vision-based detection model, we train vision data using a CNN model with transfer learning and fine-tuning methods. Simultaneously, we process sensor data with LSTM to discriminate normal drivings from abnormal ones. In the second step, we integrate the predictions of vision-based CNN and sensor-based LSTM-based models into an ultimate LSTM model and get the final predictions on the test set. We use a publicly available dataset for transfer learning. Besides, we collected a dataset comprising both vision and sensor data in the context of this study.

There are some recent works that combine multiple modalities for distracted driver detection [7,17–20]. The main contributions we bring in our work are given below:

- Applying transfer learning to vision data using two different CNN architectures,
- Using a publicly available distracted driver detection dataset to fine-tune the CNN model used to process vision data,
- Building a model to detect more distracted driving actions (ten) compared to the most of the existing studies that aim to detect six or fewer actions,
- Collecting a multimodal data from 13 real-world trips performed over 7 weeks,
- Collecting the vision and sensor data from the same source with a less-costly system by using a mobile application during the data collection process.

The rest of this paper is organized as follows. We summarize the related works in Section 2. Section 3 describes the vision and sensor data used in our study. In Section 4, we explain the details of the proposed driver action recognition system. Section 5 presents the experimental results. Finally, we present the conclusions and discussion in Section 6.

2. Related works

The works in the realm of driver action detection can be classified in three groups: vision-based, sensor-based and a combination of these two. In this section, we give a brief overview of the related studies under these three groups.

2.1. Vision-based driver action detection

In vision-based driver action detection, the previous studies mainly use images collected with a built-in camera inside the vehicle to observe actions/movements of the driver. The frames captured by the camera are classified using a machine learning algorithm such as CNN, support vector machines (SVM), k-nearest neighbors (k-NN), AdaBoost, etc.

In one work, Berri et al. [21] have identified two actions as with and without a phone and worked on the detection of these actions by using SVM on frontal images of the driver. Le et al. [22] have identified two actions as cell phone usage and hands on the wheel. They have proposed a novel method called Multi-Scale Faster R-CNN (MS-FRCNN). Their approach achieves higher accuracy with less processing time compared to the Faster R-CNN [23]. In another study, Ohn-Bar et al. [24] aimed to classify input images into one of the three actions: hands on the wheel, gear interaction, and instrument cluster (radio, etc.) interaction. They used two different cameras that are facing the hand and the face, respectively. Then, they aimed to determine the action of the driver using SVM. They obtained 90% accuracy using only hand-related information and 94% accuracy using hand and face information together. In the works of [25–29], the authors used the same dataset, that is collected by Southeast University, in which each image is labeled with one of the following actions: hands on the wheel, operating the shift lever, eating, and talking on the phone. These studies used various types of classifiers and the highest classification accuracy was achieved using CNN with 99% at [29].

In one of the recent studies, Xing et al. [30] have identified seven actions as normal driving, left mirror checking, right mirror checking, rear mirror checking, using in-vehicle radio/video device, answering the phone, and texting. They first cropped the driver's body from the images and then applied the Gaussian Mixture Model (GMM) [31] to extract the body from the background. Finally, they used different type of CNN to classify the actions. The best result was achieved by using AlexNet [32] as 91% in binary format predicting whether the driver was distracted.

In [33,34], the StateFarm's distracted driver detection dataset [35] was used, which comprises images labeled with one of the following ten actions: safe driving, texting using the right hand, talking on the phone using the right hand, texting using the left hand, talking on the phone using the left hand, operating the radio, drinking, reaching behind, hair and makeup, and talking to passengers. In [33], they first segmented driver actions using the SURF [36] key-points, then they extracted histogram of gradient features candidate regions, and finally used k-NN for classification. They finally achieved 70% recognition rate. In [34], they used two different CNN models to classify the actions. In one of these models, they used the triplet loss to improve the overall accuracy of the classification model. In this way, they have achieved an accuracy rate of 98%. We should note that they did not apply a leave-driver out cross-validation strategy.

In [37,38], the American University in Cairo (AUC) distracted driver dataset [39] was used, which is similar to the StateFarm's dataset [35]. In [37], they have trained 5 unique types of CNN using the original image, face image, hand image, face with hand image, and skin segmented image. Finally, the results obtained from CNN were weighted with the help of the genetic algorithm and the action was classified. Baheti et al. [38] applied a modified CNN with various regularization techniques to prevent over-fitting and finally they achieved 96% accuracy on the distracted driver detection task.

Behera et al. [40] used both AUC and StateFarm's distracted driver detection dataset with sequence information of the images, i.e. they used the video version of the dataset. They proposed a novel deep neural network approach called Multi-stream LSTM (M-LSTM). They used appearance features [41] and contextual information (e.g., pose and objects) obtained from another pre-trained CNN with distinct combinations as multi-stream inputs for their M-LSTM. They achieved 52.22% and 91.25% accuracy on AUC and StateFarm's distracted drivers datasets, respectively.

2.2. Sensor-based driver action detection

Sensor-based driver action detection systems generally use data collected from sensors such as gyroscope, accelerometer or devices such as GPS, OBD. The collected data is used to determine driver action with the use of various classifiers similar to vision-based driver action detection systems. In one of these studies, Ahmet et al. [42] used the driving simulator called the Computer-Assisted Rehabilitation Environment (CAREN) [43] to collect sensor data during driving. The sensor data is collected by the gyroscope and accelerometer of the mobile phone used by the driver while driving, texting, calling or reading. They then tried to determine whether the driver was distracted by giving these data as input to the random forest classifier. They could detect non-distracted driving with 98.78% accuracy, while other actions with an average accuracy of 63%.

Jafarnejad et al. [44] have aimed to find out whether the driver was distracted using only car sensor data. Some parameters of the sensor data used in this study are as follows: percentage gas pedal, steering wheel angle, vehicle speed, engine RPM, pitch, roll, yaw. They compared the results by experimenting with five different classifiers, AdaBoost, gradient boosting, random forest, k-NN and SVM, and obtained the best model using the gradient boosting classifier.

Wesley et al. [45] collected the images of the driver by using a thermal camera. Then, they aimed to determine the mental load of the driver for different actions by using these images. For this purpose, they observed skin temperature differences in the driver's forehead while performing different actions. Finally, they showed that the mental load of the driver when performing only one action (driving) was less than the mental load that occurred when trying to do two tasks (driving and talking by phone) simultaneously. They have worked on monitoring the findings rather than classification.

Bo et al. [46] have focused on determining the driver's actions using only smartphone sensors like accelerometer, gyroscope, magnetometer, etc. without any external sensors. For this purpose, they have created a Hidden Markov Model (HMM), which includes the detection of successive or interrelated actions. The actions that are attempted to be determined can be listed as: determining of entering the vehicle, determining the boarding side, determining whether it is seated in the front or back seat, determining the normal texting, and determining the texting while driving. They achieved 87.18% accuracy in detecting the action.

In [11,47], authors have made efforts to prevent the mobile phone usage to avoid distractions while driving. In [47], they aimed to determine the use of the phone while driving and prevent it with the help of a mobile jammer. First, they have developed a circuit that can detect phone usage based on energy consumption only on the driver's seat. Then, incoming and outgoing signals to the phone were prevented by using the mobile jammer while driving. In [11], they used an Arduino interface with OBD and accelerometer to collect data. This interface sends the collected data to the application on the mobile phone via Bluetooth. The application restricts incoming and outgoing calls to the phone when the speed of the vehicle exceeds 10 km/h. Also, the incoming data was evaluated according to certain threshold values, and some actions were tried to be determined while driving. These actions are normal driving, turning left, turning right, sudden change lane towards the right, sudden change lane toward the left, u-turn (leftward), sudden acceleration, and harsh brake. They analyzed the detected actions using graphs and threshold values obtained from the sensor data.

Amarasinghe et al. [10] developed a mobile application for the driver monitoring system using real-time OBD sensors data. First, they pre-processed the data collected using the mobile application. Then, using a cloud-based back-end, they aimed to determine whether the driver was reckless and if there were any driving anomalies. They used a simple formula that measures acceleration and deceleration according to certain threshold values for specific time intervals to determine the reckless driving. Finally, measurements exceeding the threshold are classified as reckless driving.

In [48,49], the actions of the driver have been identified using the smartphone sensor data. In [48], the actions such as right and left turn, right and left lane change, sudden braking, and sudden acceleration were detected using accelerometer and gyroscope data. Similar to [48,49] aimed to identify the driver actions such as lane change, sudden acceleration, and sudden deceleration using accelerometer data obtained from the smartphone. In addition to driver actions, they used the accelerometer data to determine road conditions, including bumps, potholes, rough roads, uneven roads, and smooth roads. Also, they showed these road conditions on the map by interpolating with GPS data.

Dörr et al. [50] aimed to identify the driver's actions and driving style according to the road type (dirt track, urban street, rural roads, motorways) using CAN (controller area network) bus data. First, they obtained the road type from the vehicle's navigation system. After determining the road type, they developed subsystems by taking different parameters into account varying for each road type. For example, parameters such as longitudinal acceleration, lateral acceleration, deceleration, speed and time gap were taken into consideration for rural roads. Finally, using fuzzy logic, the driving style of the driver was tried to be determined in three situations: sport, normal, comfortable. They achieved 68% correct classification rate.

In [51,52], the goal was to determine the driver behavior and driving style using sensor data. In [51], they aimed to determine whether the driver was performing normal or aggressive driving by using the accelerometer data. They have experimented with different set of features derived from accelerometer data for classification using k-NN algorithm. They achieved the best classification results using the following measurements: mean of longitudinal acceleration, mean of vertical acceleration, the median of vertical acceleration, the covariance between longitudinal and lateral acceleration, and three fifth order polynomial coefficients after inverse cumulative density histogram approximation. In [52], they aimed to identify whether the driver was driving aggressively using data such as accelerometer, gyroscope, GPS from smartphone sensors. To determine the driving style, they first determined the actions of the driver such as aggressive right and left turn, aggressive acceleration and deceleration, and aggressive lane change. As a result, they have found that the classification success rates achieved using different sensors together were higher than the classification success rates achieved using the individual sensors. They could detect aggressive actions with 97% success.

2.3. Multimodal driver action detection

There is much less work that combined data from multiple modalities for distracted driver detection task. In one of these studies, Streiffer et al. [7] used the combination of video recordings obtained with a camera and sensor data collected with a mobile phone to determine the driver's actions. They have determined 6 different actions which were normal driving, talking, texting, eating/drinking, hair and makeup, reaching. For the first three actions, image and sensor data were used together, and for the last three actions, only image data was used. For the classification process, three different deep learning techniques were used: CNN, CNN + RNN, and CNN + SVM. The highest classification rate was achieved using CNN + RNN with 87%. One of the primary differences of our work is that, while the model in [7] was built to detect 6 actions, we aim to detect 10 different actions including the normal driving and also apply fine-tuning using a publicly available distracted driver detection dataset. Besides, it is necessary to mention that in our study image and sensor (also includes OBD) data were obtained from the same source using a mobile application during the data collection process.

Du et al. [17] used three modalities, facial expression, speech data and car information to perform a binary classification of distraction. The key contribution of this paper was to integrate the speech features into the distraction detection task. We should

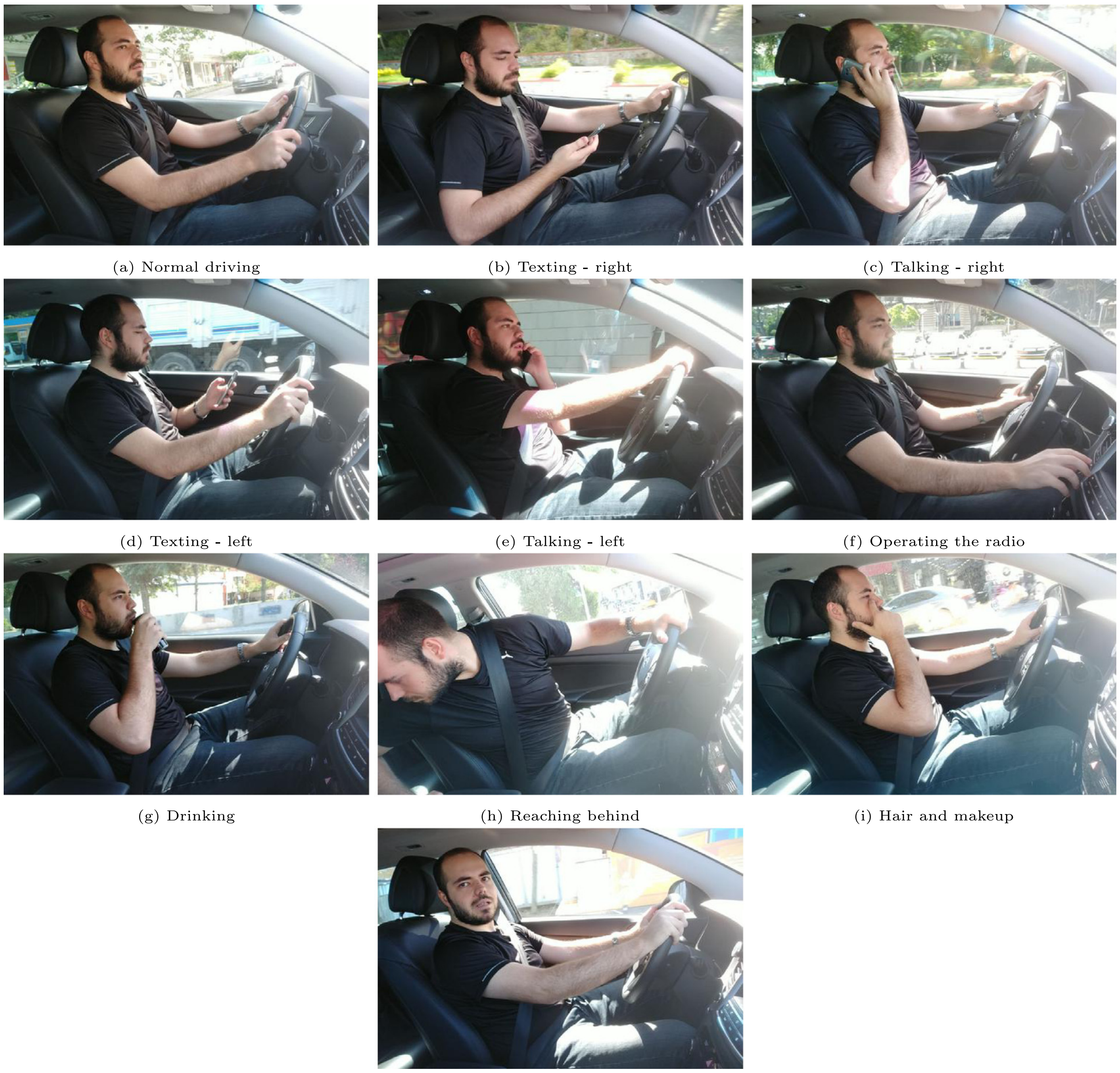

(a) Normal driving (b) Texting - right (c) Talking - right
(d) Texting - left (e) Talking - left (f) Operating the radio
(g) Drinking (h) Reaching behind (i) Hair and makeup
(j) Talking to passenger

Fig. 1. Sample images from the collected dataset.

note that the dataset in this study was collected with a driving simulator. Similarly, in [18], speech features were used along with facial expression and car information to detect the type of distraction as visual, auditory, biomechanical, and cognitive.

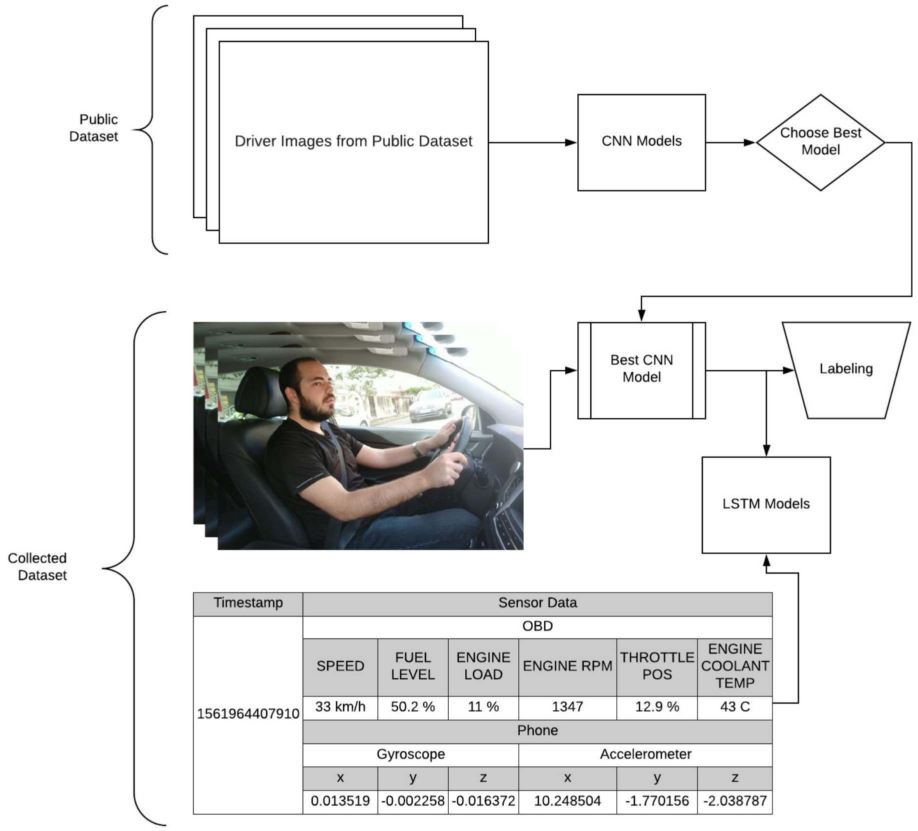
In [19], various data sources including audio, color video, depth map, heart rate, and steering wheel and pedals positions are combined for fatigue and distraction detection. They have used SVM and HMM to process data from different modules and fused the predictions of the modules using a Bayesian network. The data was collected with a driving simulator. They have built the system for two different tasks: binary classification to discriminate normal driving from distracted driving and five-class classification to detect one of the five actions which are phone call, texting, drinking, looking at an object inside the vehicle, and normal driving. In another study, [20], a real-world driving data was collected from unique data sources including frontal and road cameras, a microphone array, and a controller area network bus. The principal focus of this study was to discriminate visual and cognitive distractions and also to estimate the level of distraction rather than detecting specific action of the driver.

3. Dataset description

In our work, the driver action recognition performance is evaluated on two distinct datasets. The first dataset, called The StateFarm's distracted driver detection dataset, comprises driver images that are publicly available. The second dataset was collected in the context of our study, which comprises simultaneously taken OBD sensor data, gyroscope and accelerometer data, and driver images.

3.1. Public action recognition dataset

The StateFarm's distracted driver detection dataset published on Kaggle [35] is one of the most commonly used datasets in the related studies. Each image in StateFarm dataset is associated with one of the following driver actions: safe driving, texting using the right hand, talking on the phone using the right hand, texting using the left hand, talking on the phone using the left hand, operating the radio, drinking, reaching behind, hair and makeup, and talking to passenger. There are approximately



Timestamp	Sensor Data					
	OBD					
	SPEED	FUEL LEVEL	ENGINE LOAD	ENGINE RPM	THROTTLE POS	ENGINE COOLANT TEMP
1561964407910	33 km/h	50.2 %	11 %	1347	12.9 %	43 C
	Phone					
	Gyroscope			Accelerometer		
	x	y	z	x	y	z
	0.013519	-0.002258	-0.016372	10.248504	-1.770156	-2.038787

Fig. 2. Steps of the overall distracted driver detection system.

2200 images for each class and each image has a resolution of 640 × 480 RGB. It is important to note the following two cases about this dataset. First, the dataset contains images that are almost identical to each other as they are extracted from the same video. Second, there is no sequence information or timestamp information about the images.

3.2. Collected dataset

One of the primary goals of our study is to combine image data with the car sensor data to improve the generalization ability of the distracted driver detection model. Therefore, in addition to the public dataset, a new dataset consisting of image and sensor data has been collected. For this purpose, first, a mobile application was developed using the React Native framework. This mobile application records the video of the driver, the sensor data of the mobile phone and the OBD sensor data via Bluetooth in every 200 ms. We gathered gyroscope, accelerometer and GPS data from the mobile phone sensors and the engine load, the engine rpm, the throttle position, fuel level, engine coolant temperature, and speed information from the OBD sensors of the vehicle. The mobile phone with this application was then placed in the upper right corner of the vehicle, on the passenger side, such that the phone camera could see the driver. We chose this location since a similar placement was made in the public dataset. As a result of 13 separate trips carried out with this setting, approximately 137,000 data points with timestamp information were collected. The distribution of the data collected during these trips is shown in Table 1. Some sample frames that belong to different driver actions from the collected dataset are shown in Fig. 1.

Table 1
Number of images for each action (class) in the collected dataset.

Class code	Class name	Frame count
c0	Normal driving	105,231
c1	Texting – right	1734
c2	Talking on the phone – right	2730
c3	Texting – left	1495
c4	Talking on the phone – left	2347
c5	Operating the radio	2076
c6	Drinking	1014
c7	Reaching behind	1258
c8	Hair and makeup	5943
c9	Talking to passenger	13,265
	Total	137,093

4. Driver action recognition

We explain the details of the proposed distracted driver detection system in this section. Besides, since we propose to use vehicle sensor data together with the driver images and the vehicles have tens of sensors, the sensors that are used in the proposed system are also explained in this section.

4.1. Selection of sensors

All vehicles that are produced after the year of 1996, are required to be accessed through a standardized bus interface and each service has a special ID (OBD-II PID) that is primarily used for state-mandated emissions inspections. We examined the sensor information that is shared through the vehicle bus and find that to estimate rapid acceleration and deceleration, we should have access to speed, throttle position, engine load, and engine RPM. Since gyroscope and accelerometer data are collected in x, y and, z axes, they are represented by 3 different inputs as *accelerometer_x*, *accelerometer_y*, *accelerometer_z*, *gyroscope_x*, *gyroscope_y*, and *gyroscope_z*. Fuel level and engine coolant temperature are also collected that might help us detect dangerous actions. Apart from the car sensors, thanks to the availability of a mobile phone inside the car, we also include mobile sensor data such as gyroscope and accelerometer, which can help us detect the forward/backward motion of the car.

4.2. System overview

The proposed driver action identifying system consists of two main components. In the first part, we fine-tune various pre-trained CNN architectures using a publicly available distracted driver detection dataset, comparatively analyze the results, and apply the best CNN model to the visual part of our collected dataset that includes both the in-vehicle captured images of the driver and sensor data. Simultaneously, using the sensor data, we train an LSTM and obtain the final predictions on the test set. The steps of the proposed system are shown in Fig. 2.

In this section, the methods used to construct the system are explained. First, we give a brief review of CNN architecture and its use for transfer learning and fine-tuning for image classification tasks. We also give the pre-trained CNN models used in our study. Then, we explain the use of LSTM to process the temporal data collected in our study. Finally, we summarize the techniques used to fuse the visual and sensor data.

4.2.1. Learning from images

We create a CNN model to estimate the driver actions using in-vehicle captured images as suggested in [32]. In CNN, basic features such as lines, edges, and colors are learnt and these features are merged to detect higher-order features. CNN architecture consists of many layers that are designed for individual tasks, but we can summarize them as feature learning and classification stages.

In addition to the general structure of CNN, batch normalization [53] and dropout [54] techniques used for regularization in CNN architectures should be mentioned. We use both techniques in our work. The concept of normalization is commonly used in the machine learning area and aims to speed up the training process and eliminate excessive fluctuations by scaling input values. Batch normalization is performed similarly. However, since the input data is processed as batches, normalization is also carried out for each batch. Especially in CNN applications, batch normalization is used not only in the input layer but also in the hidden layers. This prevents the weights and outputs from reaching extremely high or extremely low values throughout the entire network. The key goal of the dropout technique is to prevent over-fitting by ignoring some randomly chosen neurons in particular epochs during training. Thus, dropout forces the network to learn more robust features and helps to improve the generalization error.

Because of the complex structure of CNN architectures, a large amount of samples is required to obtain a generalizable model. However, sometimes, there may not be a sufficient number of labeled images to learn discriminative features from raw pixels. In such cases, data augmentation methods such as horizontal and vertical flipping and rotating, changing zoom level, changing brightness are applied to the original images to increase the input data. We used data augmentation process to improve the detection accuracy of CNN in our study. We should note that data augmentation process is applied only to the training images.

Table 2
Simple CNN model.

Simple CNN model	
Block number	Layer(s)
0	Input (224 × 224) RGB image
1	Conv2D–32 with ReLu Conv2D-32 with ReLu Batch normalization MaxPooling2D Dropout(0.25)
2	Conv2D–64 with ReLu Conv2D–64 with ReLu Batch normalization MaxPooling2D Dropout(0.25)
3	Conv2D–128 with ReLu Conv2D-128 with ReLu Batch normalization MaxPooling2D Dropout(0.25)
4	Flatten Fully connected–1024 with ReLu Batch normalization Dropout(0.5) Fully connected–10 with softmax

4.3. Transfer learning and fine tuning

Transfer learning aims to use the knowledge learned from a task using huge data to solve another but similar task [55,56]. Transfer learning requires a network, pre-trained network, that is already trained for a single or multiple tasks. Then, given another similar task, the pre-trained network is utilized as a base to construct a new network to learn the new task. The latter network may be a new fully connected network appended to the end of the pre-trained network, or the features extracted from the pre-trained network can be fed to any other learning algorithm, such as SVM. In transfer learning, the hidden layer weights of the pre-trained network are not modified but only used to extract features for the new task. On the other hand, in fine-tuning, which is a variant of transfer learning, the hidden layers of the pre-trained network are also modified to improve the performance of the network further using the data belonging to the unfamiliar task.

Many improved CNN architectures have already been trained on large image datasets such as the ImageNet dataset published in ImageNet Large Scale Visual Recognition Challenges (2010–2017) [57]. The related dataset consists of approximately 1.2 million images and 1000 different classes. In this study, three different CNN architectures are used as base models as shown in Fig. 3. Two of these models, pre-trained VGG16 and Inception V3 which were applied successfully to various problems in the literature [58–63], are fine-tuned using the public distracted driver detection dataset. The third one is a simple CNN architecture built in this study for benchmark purposes. The structure of this CNN architecture is shown in Table 2. All models use RGB images with 224 width and 224 height dimensions as input.

We used Keras implementations of VGG16 and Inception V3 models consisting of 5 blocks/19 layers and 11 blocks/311 layers, respectively, for transfer learning and fine-tuning purposes. A total of 18 models with a various number of trained blocks were determined from both architectures. The details of these models are shown in Table 3. In addition to these blocks, there is a final block that consists of fully connected layers at the end of each model. This block was replaced by the last block used in the simple CNN model for our learning task.

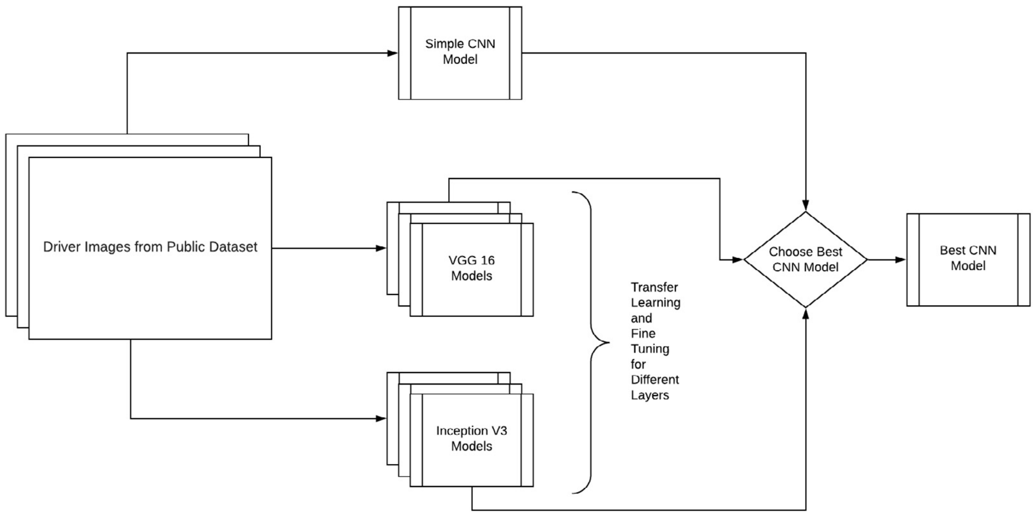

Simple CNN Model
Driver Images from Public Dataset
VGG 16 Models
Inception V3 Models
Transfer Learning and Fine Tuning for Different Layers
Choose Best CNN Model
Best CNN Model

Fig. 3. First stage of the system proposed for distracted driver detection.

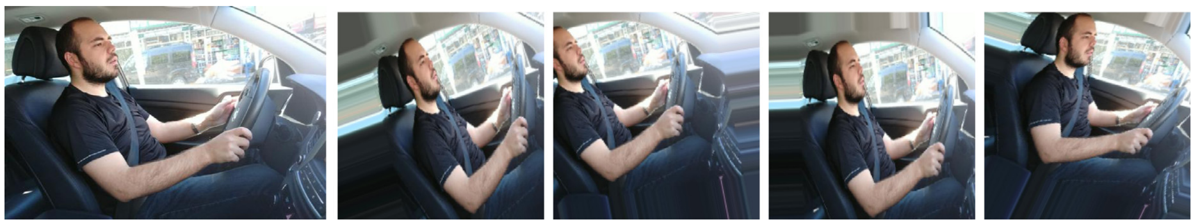


Fig. 4. Exemplary images generated by the augmentation process.

Table 3
Inception V3, and VGG16 networks fine tuning and transfer learning.

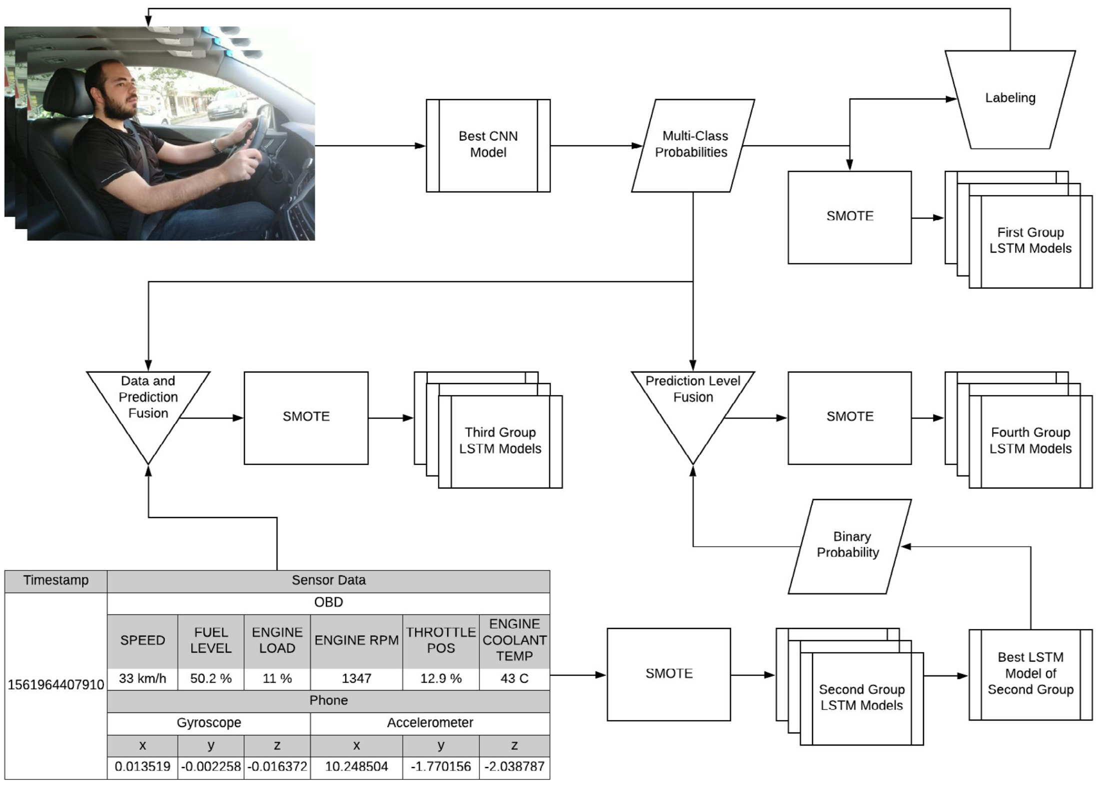
CNN model	Freezed first N block(s)	Freezed first N layer(s)	Operation
Inception V3	0	0	Fine tuning
	1	41	
	2	64	
	3	87	
	4	101	
	5	133	
	6	165	
	7	197	
	8	229	
	9	249	
	10	280	
	11	311	Transfer learning
VGG16	0	0	Fine tuning
	1	4	
	2	7	
	3	11	
	4	15	
	5	19	Transfer learning

We have determined a total of 19 different models, 18 of which are pre-trained models and 1 of which is a simple CNN model. After determining the models to be used, we applied the required pre-processing steps to prepare the data for training. As shown in Fig. 3, the training of these models has been carried out using the public action recognition dataset. To use the public dataset for training, the dataset was shuffled and the train/test split process was performed 10 times for statistical significance. The split process was first performed as 80% train and 20% test, and then the training set was divided again into two parts as 80% training and 20% validation for hyper-parameter optimization. However, since the dataset is extracted from the video recordings and especially the sequential images are very similar to each other, a leave-driver-out procedure is applied for cross-validation to construct an unbiased model. For this purpose, the data that contains images belonging to 26 different drivers are divided into two parts so that the training set contains the images of 20 randomly selected drivers and the test set contains the images of the remaining 6 drivers. Then, the data augmentation process was performed on the training set. The augmentation process was also applied to the collected data set and some examples obtained as a result of this process are shown in Fig. 4.

The training process was carried out separately for each model. In performing this process, the models were trained for 40 epochs using the Adam optimizer [64]. Adam optimizer aims to learn the problem faster by optimizing or adapting the learning rate. Each model was validated using the previously separated validation set. The model with the highest validation accuracy was evaluated on the test set. The CNN training operations were performed using 1 GPU and 61 GB ram on a cloud service.

After the completion of the training process, the best CNN structure was determined according to the average accuracy value of the test sets. For the best CNN structure, a final model has been trained using the specified hyper-parameters and using the entire dataset for training. With the determination of the final model, the second stage of the system was implemented as shown in Fig. 5. For the second part, within the context of the transfer learning procedure, we first applied the obtained CNN model to the collected unlabeled dataset and obtained the labels. We should note that this model is not 100% successful in classifying images. For this reason, after the transfer learning step, a manual labeling procedure has been performed.

The CNN model gives a probability that the image may belong to for each class. For manual labeling, first, the threshold values



Timestamp	Sensor Data					
	OBD					
	SPEED	FUEL LEVEL	ENGINE LOAD	ENGINE RPM	THROTTLE POS	ENGINE COOLANT TEMP
1561964407910	33 km/h	50.2 %	11 %	1347	12.9 %	43 C
	Phone					
	Gyroscope			Accelerometer		
	x	y	z	x	y	z
	0.013519	-0.002258	-0.016372	10.248504	-1.770156	-2.038787

Fig. 5. Second stage of the proposed distracted driver detection system.

for each class were determined using the probabilities obtained by the CNN model. To determine the threshold values, the images of each class determined by CNN were divided into 5% intervals according to probability estimation values and grouped according to these intervals. Afterwards, these groups were sorted in descending order according to their probability estimates. For each interval, 2 images were randomly selected for each of the 13 trips and examined manually whether at least 23 of these 26 randomly selected images were classified correctly. If the model performance is less than the predetermined threshold, the probability value of the previous group is considered as the threshold value for this class. If the classification is successful, the next group was processed and the above steps were repeated until the classification failed. In this way, after determining the threshold value for each class, the images with the probability below this threshold value were separated for manual classification. For this purpose, an interface for manual labeling was developed as shown in Fig. 6. The images to be labeled were sorted by timestamp to ease the labeling process and the images were labeled using the developed interface.

4.4. LSTM-RNN for sequential learning

In the second part of the study, we aim to process the image and sensor data sequentially. We first process the image and sensor data individually. Then, in line with the main goal of this study, we fuse them to detect the action of the driver more accurately. Since the timestamp information is available in the collected dataset, we used LSTM-RNN to model the dataset sequentially.

RNNs can model the sequential information using some additional weights between information at time *t-1* and *t*. However, it has been shown that RNN is not very successful to model the long-term dependencies in the data [65]. As the time between the inputs increases, the error vanishes throughout the network layers due to the internal structure of RNN. This is called the vanishing gradient problem. LSTM networks were introduced by Hochreiter and Schmidhuber [66] to model long-term dependencies by replacing the traditional network units with special structures called memory cell. Today, LSTM has been successfully implemented where it is necessary to use sequential information like speech recognition, speech synthesis, sentiment analysis, handwriting recognition, click-stream analysis, question answering, etc. [67–72].

In this study, many different LSTM models have been implemented as shown in Fig. 5. These models can be grouped according to the data they use as input. Multiple LSTM models have been implemented using image and sensor data individually or their combinations in various ways. For each of the constructed model, the dataset was shuffled and the train/test split process was performed 10 times as in the training of CNN models. The split process was first performed as 80% train and 20% test. For hyper-parameter optimization, 80% of the train set was used for training and the rest for validation. Unlike the public dataset, since the collected dataset is imbalanced, as seen in Table 1, the split operation was performed on a class basis. In this way, the class distribution of the dataset was preserved within each split. After completing the split process, SMOTE [73] was applied to the training set to eliminate the imbalance in the dataset. Different window sizes from 1 to 90 were tried during the training of LSTM models. All LSTM models were trained for 100 epochs.

The first group of LSTM models was implemented using only the image data. We should note that not the original images but the class probability estimates obtained by applying the best CNN model to the images were used to represent image information in LSTM models. The primary reason for this is that using images as input is much more costly than using class probabilities. Using an image as input requires 50176 (224×224) entries, whereas using class probabilities as input requires only 10 entries. All LSTM models used throughout the study were constructed with the

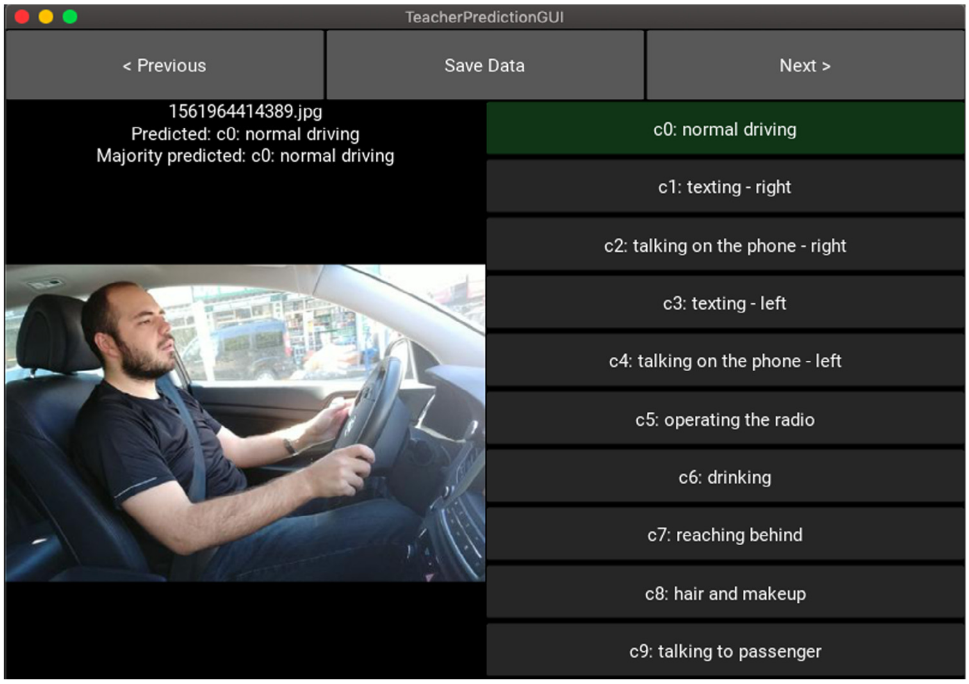


Fig. 6. Manual Labeling Interface.

Table 4
Structure of all LSTM models used throughout the study unless otherwise noted.

LSTM models layers
Input (Different window sizes × 10 Class probabilities)
LSTM-32
Dropout(0.3)
Dense-10 with softmax

Table 5
Structure of the LSTM Model that uses the sensor data as input.

LSTM models layers
Input (Different window sizes × 12 sensor features)
LSTM-32
Dropout(0.3)
Dense-1 with sigmoid

structure shown in Table 4 for the input layer unless otherwise noted.

The second group of LSTM models was implemented using only the sensor data. However, several cases have been identified as some OBD sensor data is missing. For example, when the engine is started for the first time, the data received from the OBD may be corrupted or incomplete. Such missing or incorrect sensor recordings have been removed from the dataset. In addition to the pre-processing operations applied for the image data, we have also applied standardization to rescale the sensor features.

An important point to be noted here is that the models created with sensor data were designed to generate a binary output representing whether the driver's action is normal driving. In other words, the other 9 classes were not taken into account while building the LSTM model using only sensor data since using only the sensor data was not sufficient to distinguish between driver actions but only can give useful information in distinguishing normal driving actions from the abnormal ones. Therefore, the binary outputs generated by this model were used by fusing it with the image information. The sensor-based LSTM models were implemented as shown in Table 5. With the completion of the training process, the model with the highest validation accuracy was chosen as the best model of this group and the binary outputs obtained using this model were stored for later use.

4.5. Fusion of sensor and image-based models

The last step of the proposed methodology is to combine the individual prediction models built with sensor and image datasets as seen in Fig. 5. The main goal of combining the information coming from these two data sources is to improve the generalization ability of the prediction model. Before giving the details of the fusion techniques used at this step, it is necessary to mention the process of synchronizing image and sensor data. The steps required for this process have been completed in the mobile application side implemented during data collection. The data recording process was triggered every 200 ms during data collection. At each trigger, both the image and the sensor data were recorded using the same timestamp. In this way, the collected image and sensor data were matched to each other using the timestamp information. The only process for preparing the dataset used in this stage was to remove images with the same timestamp as the removed missing sensor data.

As seen in Fig. 5, we used two different fusion techniques to combine sensor and image data. The first one is a hybrid version of data and prediction level fusion techniques which merges sensor data and multi-class probability estimates obtained from the image-based model. After integrating this information, we have applied standardization as a pre-processing step to rescale the sensor features and fed the merged dataset into the LSTM model shown in Table 6.

As the second fusion technique, we used a prediction level fusion approach in which the binary probability estimates of the sensor-based model and multi-class probability estimates of the image-based model are merged. Thus, we aim to determine the better way of integrating the sensor information into the combined model. For prediction level fusion, we used the same LSTM architecture with that of hybrid fusion shown in Table 6, except the input layer.

Table 6
LSTM model constructed for hybrid fusion.

LSTM models layers
Input (Different window sizes × (10 class probabilities + 12 Sensor features))
LSTM-32
Dropout(0.3)
Dense-10 with softmax

Table 7
Average test set accuracies of the CNN Models on public distracted driver detection dataset.

CNN model	Freezed first N block(s)	Freezed first N layer(s)	Operation	Average test accuracy
Simple CNN	0	0		**0.7235**
Inception V3	0	0	Fine tuning	**0.7976**
	1	41		0.4058
	2	64		0.7014
	3	87		0.6469
	4	101		0.5986
	5	133		0.5544
	6	165		0.7016
	7	197		0.6500
	8	229		0.5557
	9	249		0.5329
	10	280		0.3286
	11	311	Transfer learning	0.3239
VGG16	0	0	Fine tuning	0.7514
	1	4		**0.7694**
	2	7		0.7376
	3	11		0.7341
	4	15		0.6928
	5	19	Transfer learning	0.6110

5. Experimental results

In this section, the results obtained in each step of the proposed distracted driver detection system shown in Fig. 2 are given comparatively. The results are presented in the order of the application of the methods. For evaluating our system, we aim to answer the following questions:

- How does fine-tuning the pre-trained CNN models affect the performance of the image-based detection model?
- To what extent does the sensor data improve the image-based detection model?
- What is the suitable way of combining image and sensor data?

5.1. Image-based prediction results on public dataset

We first give the results of CNN models that are trained using only the public distracted driver detection dataset. As mentioned in Section 4, a total of 19 CNN models were applied to the public dataset. The train/test split process was repeated 10 times and the hyper-parameter optimization is performed on the training set by splitting it into training and validation sets. We performed a non-parametric statistical test, Wilcoxon test, between each pair of classifiers to examine whether there is a significant difference between the test set accuracies of different models.

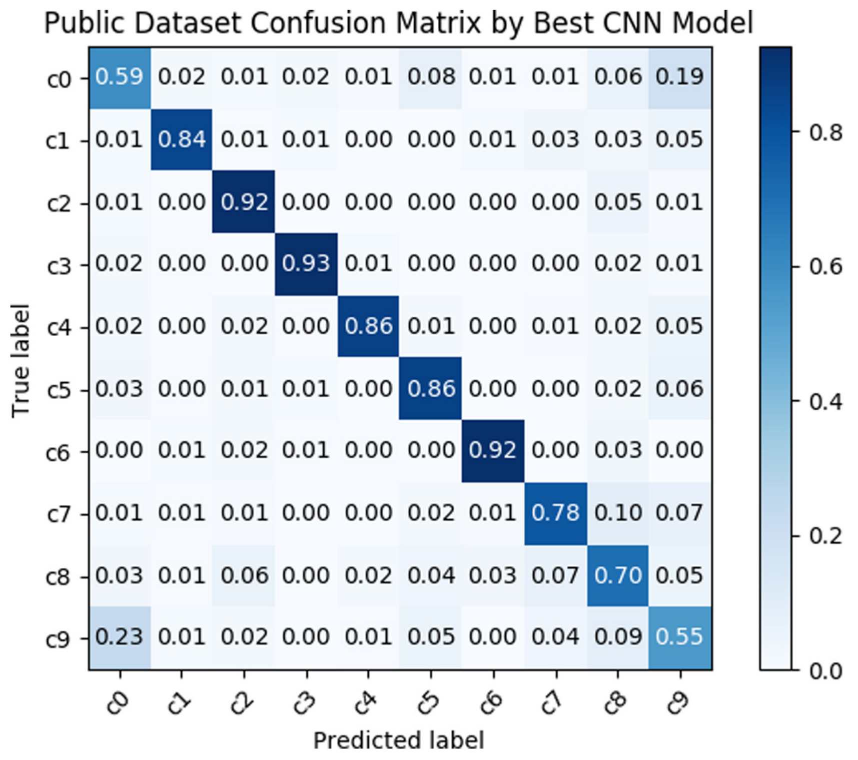
The average test set accuracies are shown in Table 7. As seen, the simple CNN model built in the context of our study without transfer learning gave 72% accuracy. This result can be considered as a baseline for the pre-trained network-based CNN models. It is seen that both Inception V3 and VGG16 models achieved higher accuracies than simple CNN model showing that using a pre-trained model with fine-tuning improves the generalization ability of the distracted driver detection model.

As seen in Table 7, the highest accuracy was obtained using a fine-tuned Inception V3 model with 79.76%. The Wilcoxon statistical test showed that the performance of this model is significantly better than simple CNN and best VGG16 model (p-value < 0.01). We should also note that fine-tuning, in general, significantly increased the accuracy of both Inception V3 and VGG16 models. The best Inception V3 model was obtained by fine-tuning all layers. On the other hand, we see that the highest accuracy with VGG16 was obtained by freezing the first 4 layers and fine-tuning the rest. However, the Wilcoxon test revealed that the difference between the accuracies of VGG16 and simple CNN is not significant (p-value > 0.05). Another point to be noted is that while pure transfer learning, i.e. re-training none of the layers, based Inception V3 showed poor performance with 32.39% classification accuracy, transfer learning-based VGG16 gave 61% accuracy. These findings show that although the best performance is obtained with the Inception V3 model, VGG16 features extracted without fine-tuning fit better with the distracted driver detection problem than Inception V3 features.

Based on the obtained results, the Inception V3 model with fine-tuning has been chosen to be applied to the image part of the unlabeled action detection data collected in this study. The confusion matrix showing the class-based test set accuracies obtained by applying this model on the public dataset is shown in Fig. 7. The results show that the accuracy of the model in detecting normal drivings is 59%. We also see that the model performs very well in the detection of $c2$, talking on the phone — right, $c3$, texting — left, and $c6$, drinking, actions achieving above 90% accuracy. On the other hand, the lowest class-based accuracy is obtained for $c9$, talking to a passenger. This action is mostly confused with $c0$, normal driving, and $c8$, hair-and-makeup actions.

5.2. Image-based prediction results on collected dataset

As mentioned in Section 4, we utilize the best CNN model to reduce the effort in labeling the images of the dataset collected in the context of our study. For this purpose, first, we have applied



	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9
c0	0.59	0.02	0.01	0.02	0.01	0.08	0.01	0.01	0.06	0.19
c1	0.01	0.84	0.01	0.01	0.00	0.00	0.01	0.03	0.03	0.05
c2	0.01	0.00	0.92	0.00	0.00	0.00	0.00	0.00	0.05	0.01
c3	0.02	0.00	0.00	0.93	0.01	0.00	0.00	0.00	0.02	0.01
c4	0.02	0.00	0.02	0.00	0.86	0.01	0.00	0.01	0.02	0.05
c5	0.03	0.00	0.01	0.01	0.00	0.86	0.00	0.00	0.02	0.06
c6	0.00	0.01	0.02	0.01	0.00	0.00	0.92	0.00	0.03	0.00
c7	0.01	0.01	0.01	0.00	0.00	0.02	0.01	0.78	0.10	0.07
c8	0.03	0.01	0.06	0.00	0.02	0.04	0.03	0.07	0.70	0.05
c9	0.23	0.01	0.02	0.00	0.01	0.05	0.00	0.04	0.09	0.55

Fig. 7. Average confusion matrix of the best CNN model obtained on the test set of the public dataset.

Table 8
Threshold values used for manual labeling process of images in the collected dataset.

Class code	Class name	Threshold
c0	Normal driving	30%
c1	Texting – right	0%
c2	Talking on the phone – right	90%
c3	Texting – left	90%
c4	Talking on the phone – left	100%
c5	Operating the radio	65%
c6	Drinking	95%
c7	Reaching behind	85%
c8	Hair and makeup	100%
c9	Talking to a passenger	95%

the best CNN model, Inception V3 model with fine-tuning, to the images of the collected dataset. Then, we have determined the threshold values of the probability estimates for each class with the methodology detailed in Section 4. Finally, we have used these threshold values to determine the images that will be manually labeled. The remaining images on which the CNN model is confident have not been included in the manually labeling process and the predictions of the CNN model have been directly used as the grand truth for these images. The obtained threshold values are shown in Table 8.

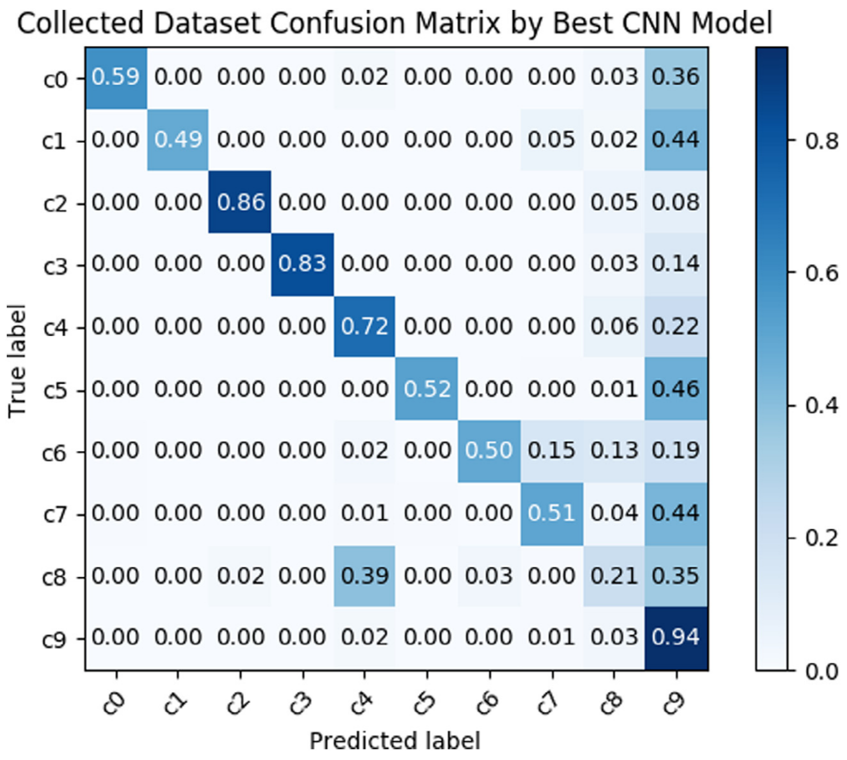
When the threshold values determined in Table 8 are examined, the first thing to be noted is that the threshold values are generally high. We also see that the model is successful in classifying $c0$, normal driving, and $c1$, texting with the right hand, actions. The threshold values of the other actions indicate that the model misclassified the actions which include regionally close movements to each other. For example, $c5$, operating radio and $c7$, reaching behind actions include similar movements. Apart from all these, we have also observed that the model trained with the public dataset tends to classify the samples of the collected dataset as $c9$, talking to a passenger, due to the difference between the camera angles of the public dataset and the collected dataset.

After the semi-automatic labeling process, the collected dataset was shuffled and the train/test split process was performed 10 times similar to the experiments performed on the public dataset. The Inception V3 model detected the actions with an average accuracy of 62%. The confusion matrix showing the average class-based accuracies obtained on the test set is given in Fig. 8. The comparative analysis of the results obtained on public and collected image datasets point out that although the normal driving action detection accuracies are very close, 59%, the overall success rate on the collected dataset is much lower.

5.3. LSTM-RNN results on collected image and sensor data

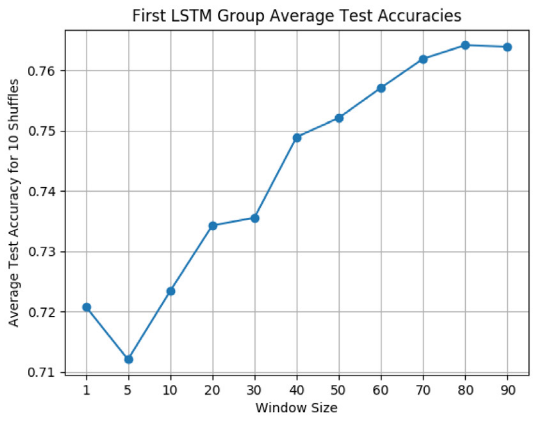
In the second part of the study, various LSTM networks were built to process the collected data sequentially as mentioned in Section 4. The first LSTM model uses class probabilities produced by the best CNN model as input. We used the same cross-validation procedure applied to the image dataset. The mean accuracies obtained with varying window sizes from 1 to 90 are given in Fig. 9. As seen, the highest mean test accuracy with 76% was obtained using 80 as the window size. We have observed that after this point increasing window size does not improve the accuracy significantly. The confusion matrix of the predictions on the test set is shown in Fig. 9.

When the results obtained using the LSTM model are examined, it is seen that an average of 14% improvement from 62% to 76% was achieved compared to the results obtained with CNN. The main reason for this is that LSTM can process not only the information in the current image but also the information in the preceding images. When the confusion matrices of CNN and LSTM models given in Fig. 8 and Fig. 9, respectively, are examined, it is seen that a significant improvement has been achieved with LSTM in the classification of all actions except $c9$, talking to a passenger,

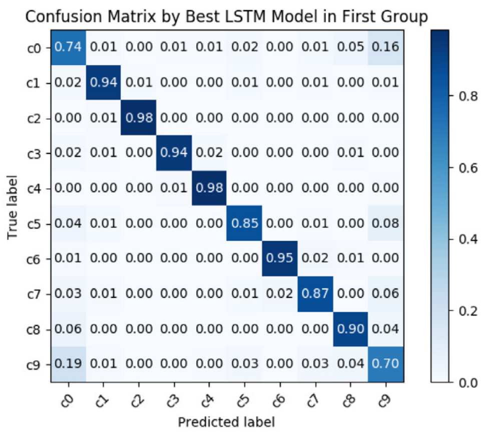


	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9
c0	0.59	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.03	0.36
c1	0.00	0.49	0.00	0.00	0.00	0.00	0.00	0.05	0.02	0.44
c2	0.00	0.00	0.86	0.00	0.00	0.00	0.00	0.00	0.05	0.08
c3	0.00	0.00	0.00	0.83	0.00	0.00	0.00	0.00	0.03	0.14
c4	0.00	0.00	0.00	0.00	0.72	0.00	0.00	0.00	0.06	0.22
c5	0.00	0.00	0.00	0.00	0.00	0.52	0.00	0.00	0.01	0.46
c6	0.00	0.00	0.00	0.00	0.02	0.00	0.50	0.15	0.13	0.19
c7	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.51	0.04	0.44
c8	0.00	0.00	0.02	0.00	0.39	0.00	0.03	0.00	0.21	0.35
c9	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.01	0.03	0.94

Fig. 8. Average confusion matrix of the best CNN model on the test sets of the collected dataset.



(a) Average accuracy for 10 shuffles



	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9
c0	0.74	0.01	0.00	0.01	0.01	0.02	0.00	0.01	0.05	0.16
c1	0.02	0.94	0.01	0.00	0.00	0.01	0.00	0.01	0.00	0.01
c2	0.00	0.01	0.98	0.00	0.00	0.00	0.00	0.00	0.00	0.00
c3	0.02	0.01	0.00	0.94	0.02	0.00	0.00	0.00	0.01	0.00
c4	0.00	0.00	0.00	0.01	0.98	0.00	0.00	0.00	0.00	0.00
c5	0.04	0.01	0.00	0.00	0.00	0.85	0.00	0.01	0.00	0.08
c6	0.01	0.00	0.00	0.00	0.00	0.00	0.95	0.02	0.01	0.00
c7	0.03	0.01	0.00	0.00	0.00	0.01	0.02	0.87	0.00	0.06
c8	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.90	0.04
c9	0.19	0.01	0.00	0.00	0.00	0.03	0.00	0.03	0.04	0.70

(b) Confusion matrix

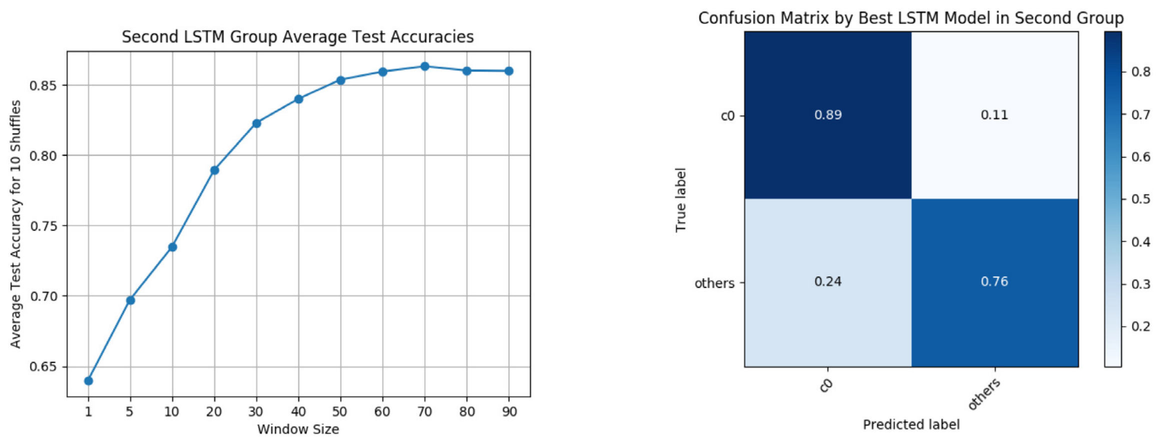
Fig. 9. (a) Average accuracy and (b) average confusion matrix of the best LSTM model on the test sets using only class probabilities of the image data as input.

action. It has been observed that normal driving accuracy is also improved from 59% to 74%. The highest improvement from 21% to 90% was observed in the detection of *c*8, hair and makeup action. Contrary to other actions, it is seen that there is a decrease in the detection accuracy of *c*9, talking to a passenger, action from 94% to 70%. In particular, we see that this action was often confused with normal driving action by the LSTM model.

The second LSTM model was based on the use of sensor data individually as input. This model, as detailed in Section 4, is designed to produce a binary output to determine whether the action is normal or not. When the results in Fig. 10 are examined, it is seen that this model is more successful in detecting normal driving than abnormal actions. As in the image-based LSTM model, the training process was carried out using different window sizes and 70 has been found as the optimal window size as seen in Fig. 10. The highest average test set accuracy obtained using only sensor data was 86%. We should note that this accuracy is not comparable to that of previous CNN and LSTM models since the action detection problem in this model was reduced to a binary classification problem, normal or abnormal driving, from a 10-class classification problem. The main purpose of this LSTM model was not to detect the specific action of the driver but to produce intermediate outputs that can be used as input in the later steps of the study.

5.4. LSTM-RNN results obtained by fusing image and sensor data

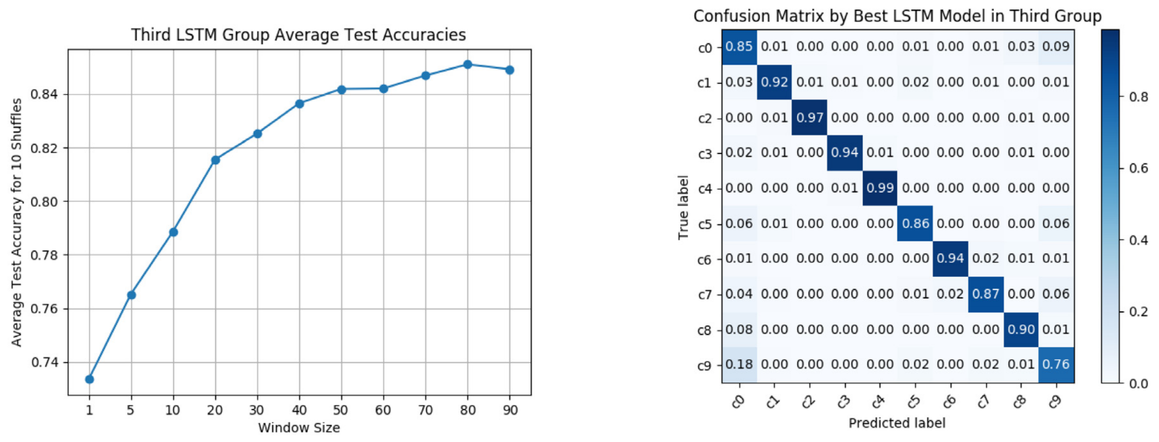
As detailed in Section 4, we have fused image and sensor data in two different ways. In the first approach, data level and prediction level fusion techniques are used in a hybrid way by merging raw sensor data and multi-class probability estimates obtained on image data. The same cross-validation procedure applied for



(a) Average accuracy for 10 shuffles

(b) Confusion matrix

Fig. 10. (a) Average accuracy and (b) average confusion matrix of the best LSTM model on the test sets using only sensor data as input.



(a) Average accuracy for 10 shuffles

(b) Confusion matrix

Fig. 11. (a) Average accuracy and (b) average confusion matrix of the best LSTM model on the test sets using the combination of sensor data and multi-class probabilities of the images as input.

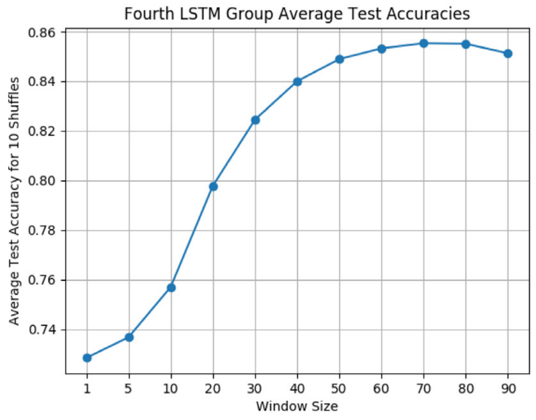
the previous CNN and LSTM models was followed. Fig. 11 shows that the highest average test set accuracy with 85% was obtained using 80 as the window size. The Wilcoxon test performed on the test set accuracies of the first and second LSTM models indicated that the second LSTM model performed significantly better than the first LSTM model that uses only class probabilities obtained on the image data with CNN model (p-value < 0.01). The test set confusion matrix belonging to this model given in Fig. 11 shows that normal driving detection accuracy is increased to 85% in this model. We also see an increase in the detection of $c9$ action from 70% to 76% when compared to the LSTM model that uses only the CNN probabilities as input.

The second attempt to fuse the sensor and image data is to use a fully prediction level fusion approach. We combine multi-class probability estimates obtained on image data with the best CNN model and binary output obtained on sensor data using the best LSTM model. The results of the prediction level fusion are given in Fig. 12. It is seen that the highest average test set accuracy with 85% was obtained using 70 as the window size. The confusion matrix of the corresponding model is shown in Fig. 12.

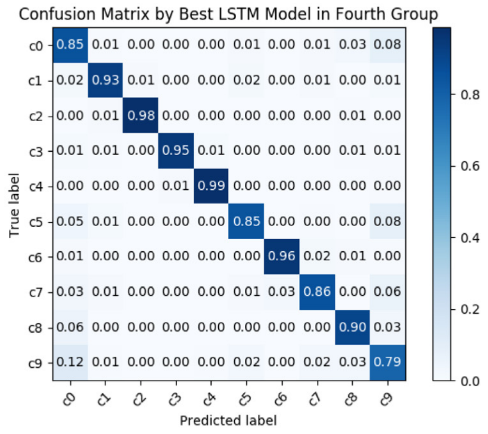
5.5. Summarized results

We give the summarized class-based and overall test set results obtained on the collected dataset in Table 9. Besides, we give the sensitivity of each model when the problem is considered as a binary classification problem in which normal driving is discriminated from all other distracted driving actions. As it is seen, the highest overall accuracies were obtained as 85% with the two LSTM models that fuse image and sensor data. Compared to the results obtained using image data with CNN, an overall significant (p-value < 0.01) improvement of approximately 23% was achieved with fused-data based LSTMs. We also see that there is an increase of 9% compared to the LSTM model built using only image data. The Wilcoxon test results showed that the difference between the overall accuracies of image-based LSTM and both fused-data based LSTMs is significant (p-value < 0.01). The binary class sensitivities of the models with sensor data are also significantly higher than that of the models without sensor data (p-value < 0.01). These findings indicate that integrating sensor data to the detection system increases the overall distracted driver detection accuracy and sensitivity. The results in Table 9 also show that the performances of the fused LSTM models are very close to each other.

As seen in Figs. 9–12, the accuracy in LSTM models is increasing in the window size up to around 70–80 and then remains almost stable afterwards. This optimal window size is related to the maximum duration that a distracting action in the collected data lasts. Our further analysis showed that, among the 9 distracted actions covered in the detection model, the hair and



(a) Average accuracy for 10 shuffles



(b) Confusion matrix

Fig. 12. (a) Average accuracy and (b) average confusion matrix of the best LSTM model on the test sets using the combination of binary sensor probability and multi-class probabilities of the images as input.

Table 9
Summarized class-based and overall test set results obtained on collected dataset.

	Without sensor data		With sensor data	
	Fine-tuning CNN	Image-based LSTM	Hybrid fusion LSTM	Fully prediction level fusion LSTM
$c0$	0.59	0.74	**0.85**	**0.85**
$c1$	0.49	**0.94**	0.92	0.93
$c2$	0.86	**0.98**	0.97	**0.98**
$c3$	0.83	0.94	0.94	**0.95**
$c4$	0.72	0.98	**0.99**	**0.99**
$c5$	0.52	0.85	**0.86**	0.85
$c6$	0.50	0.95	0.94	**0.96**
$c7$	0.51	**0.87**	**0.87**	0.86
$c8$	0.21	**0.90**	**0.90**	**0.90**
$c9$	**0.94**	0.70	0.76	0.79
Overall accuracy	0.62	0.76	**0.85**	**0.85**
Normal vs Distracted sensitivity	0.69	0.78	0.86	**0.87**

makeup action takes the maximum time in average (5 to 20 s) and this duration corresponds to 25 to 100 frames (5 fps used during recording) which is very close to the optimal window size obtained in our experiments. Therefore, increasing window size beyond these values does not bring additional information for the LSTM model.

6. Conclusions and discussion

In this work, we fused in-vehicle sensor information with the images of the driver taken while driving to increase the correct classification rate of the distracted driver detection task. We have used a publicly available dataset with transfer learning and fine-tuning methods to build a highly accurate vision-based detection model. Besides, we built a data collection application that accesses to in-vehicle sensor data through a CAN bus interface and collected a multimodal dataset by performing real-world trips. We have applied the model obtained on the public dataset to the vision part of our multimodal dataset. Finally, we combined the predictions of vision and sensor-based modalities with two different fusion approaches to map the inputs to one of the ten actions.

An important finding of this paper is to show that fusing the sensor data with vision data increases the accuracy of distracted driver detection task significantly. Specifically, both hybrid and prediction level fusion increased the overall accuracy by 9%, from 76% to 85%, when compared to using only image data. We should also note that using sensor data increased the normal driving detection accuracy from 74% to 85%. The statistical tests showed that both of these differences are significant. Considering that a low false-positive ratio is an important requirement of such user assistance systems, showing that the false-positive ratio can be decreased by integrating sensor information into the vision-based detection system can be regarded as an important contribution of our study. The experiments showed that hybrid and fully prediction level fusion-based LSTM give similar accuracies, 85%, on the collected dataset and the statistical test showed that the difference between their overall accuracies is not significant. The results also indicated that hybrid and prediction level fusion techniques performed similarly with 85% accuracy in detecting the normal driving actions. However, prediction level fusion gave a higher accuracy of 3% than hybrid fusion in the detection of $c9$, talking to passenger, action. We should note that this is the highest difference between the class-based accuracies of the fused LSTM. For the other actions, the difference never exceeded 2%. Therefore, we can conclude that the full prediction level fusion based-LSTM in which binary class predictions of the sensor-based model and multi-class predictions of the vision-based model are combined can be preferred in the final distracted driver detection system.

Another contribution of our study was to improve the accuracy of the vision-based model using the fine-tuning method. Some previous works have already applied transfer learning for the distraction detection task. These studies used a pre-trained CNN model to detect distractions on their dataset. In our study, we perform fine-tuning on a pre-trained CNN model with the usage of a public distracted driver detection dataset before applying it to our dataset. This is important to show to what extent a model built on a dataset can be generalized to a different dataset collected with different settings under different conditions. The results showed that while the pure transfer learning accuracies of the VGG16 and Inception V3-based CNN models on the public dataset were 61% and 32%, respectively, they have increased to 77% and 80% with fine-tuning.

The highest test set accuracy on the public dataset was obtained using the Inception V3 model. However, we should note that when this CNN model has been applied to the imaging modality of the collected dataset, the accuracy declined to 62%. The main reason for this performance loss was mostly because the collection setup such as the viewpoint of the camera and car model was not the same as that of the public dataset. For example, the findings showed that the normal driving action in the collected dataset is mostly classified as 'talking to a passenger' action by the vision-based CNN model due to the difference in camera angle between the data collection setup of the public and collected datasets. While in the collected dataset the driver's face is facing slightly more towards the camera in almost all of the actions, in the public dataset the driver's face is facing towards the camera only when talking to the passenger.

Many vehicle apps may benefit from driver-specific functions, and the number of these apps will only increase with increasing access to in-vehicle data. The following examples are considered with this assumption. Automotive insurance companies are calculating the monthly premiums by not just using the number of accidents of the driver involved with, but also using the sensor information such as rapid acceleration, hard braking, etc. If the driver actions can be classified accurately while driving, this information can be used as a parameter of insurance policies by such insurance companies. We should note that camera placement inside a vehicle and being recorded can be considered as a privacy concern for some drivers. In the proposed system, although video recording of the driver is used for action detection, this information can be processed in real-time without being stored in the database to produce probability estimations for distracted actions. A limitation of our study is that although the public dataset used for the transfer learning part includes images of multiple drivers, the collected dataset includes images belonging to only one real driver. As a future work, the reliability of the obtained results can be increased by collecting data from more drivers. Especially, collecting data from multiple drivers with various demographic information and driving style may help to improve the generalization ability of the distracted action detection model.

CRediT authorship contribution statement

Furkan Omerustaoglu: Methodology, Software, Formal analysis, Investigation, Data curation, Writing - original draft, Visualization. **C. Okan Sakar:** Conceptualization, Methodology, Writing - review & editing, Supervision, Project administration. **Gorkem Kar:** Conceptualization, Methodology, Validation, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] P.F. Ehrlich, B. Costello, A. Randall, Preventing distracted driving: A program from initiation through to evaluation, Am. J. Surg. 219 (6) (2020) 1045–1049.

[2] M.Q. Khan, S. Lee, A comprehensive survey of driving monitoring and assistance systems, Sensors 19 (11) (2019) 2574.

[3] V. Manzoni, A. Corti, P. De Luca, S.M. Savaresi, Driving style estimation via inertial measurements, in: 13th International IEEE Conference on Intelligent Transportation Systems, IEEE, 2010, pp. 777–782.

[4] A. Corti, C. Ongini, M. Tanelli, S.M. Savaresi, Quantitative driving style estimation for energy-oriented applications in road vehicles, in: IEEE International Conference on Systems, Man, and Cybernetics, IEEE, 2013, pp. 3710–3715.

[5] S.A. Birrell, M.S. Young, The impact of smart driving aids on driving performance and driver distraction, Transp. Res. F: Traffic Psychol. Behav. 14 (6) (2011) 484–493.

[6] G.M. Fitch, S.A. Soccolich, F. Guo, J. McClafferty, Y. Fang, R.L. Olson, M.A. Perez, R.J. Hanowski, J.M. Hankey, T.A. Dingus, The Impact of Hand-Held and Hands-Free Cell Phone Use on Driving Performance and Safety-Critical Event Risk, Technical Report, U.S. Department of Transportation, National Highway Traffic Safety Administration, 2013.

[7] C. Streiffer, R. Raghavendra, T. Benson, M. Srivatsa, Darnet: A deep learning solution for distracted driving detection, in: Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference: Industrial Track, ACM, 2017, pp. 22–28.

[8] F. Vicente, Z. Huang, X. Xiong, F. De la Torre, W. Zhang, D. Levi, Driver gaze tracking and eyes off the road detection system, IEEE Trans. Intell. Transp. Syst. 16 (4) (2015) 2014–2027.

[9] Q. Ji, X. Yang, Real-time eye, gaze, and face pose tracking for monitoring driver vigilance, Real-Time Imaging 8 (5) (2002) 357–377.

[10] M. Amarasinghe, S. Kottegoda, A.L. Arachchi, S. Muramudalige, H.D. Bandara, A. Azeez, Cloud-based driver monitoring and vehicle diagnostic with OBD2 telematics, in: Fifteenth International Conference on Advances in ICT for Emerging Regions, ICTer, IEEE, 2015, pp. 243–249.

[11] A. Khandakar, M.E. Chowdhury, R. Ahmed, A. Dhib, M. Mohammed, N.A. Al-Emadi, D. Michelson, Portable system for Monitoring and controlling driver behavior and the use of a mobile phone while driving, Sensors 19 (7) (2019) 1563.

[12] G. Kar, S. Jain, M. Gruteser, J. Chen, F. Bai, R. Govindan, PredriveID: Pre-trip driver identification from in-vehicle data, in: Proceedings of the Second ACM/IEEE Symposium on Edge Computing, ACM, 2017, p. 2.

[13] F. Martinelli, F. Mercaldo, V. Nardone, A. Orlando, A. Santone, Who's driving my car? A machine learning based approach to driver identification, in: Proceedings of the 4th International Conference on Information Systems, Security and Privacy, ICISSP, 2018, pp. 367–372.

[14] M. Martínez, J. Echanobe, I. del Campo, Driver identification and impostor detection based on driving behavior signals, in: IEEE 19th International Conference on Intelligent Transportation Systems, ITSC, IEEE, 2016, pp. 372–378.

[15] Z. Xiao, Z. Hu, L. Geng, F. Zhang, J. Wu, Y. Li, Fatigue driving recognition network: Fatigue driving recognition via convolutional neural network and long short-term memory units, IET Intell. Transp. Syst. 13 (9) (2019) 1410–1416.

[16] X. Li, L. Hong, J.-c. Wang, X. Liu, Fatigue driving detection model based on multi-feature fusion and semi-supervised active learning, IET Intell. Transp. Syst. 13 (9) (2019) 1401–1409.

[17] Y. Du, C. Raman, A.W. Black, L.-P. Morency, M. Eskenazi, Multimodal polynomial fusion for detecting driver distraction, in: Proceedings of the Interspeech Conference, 2018, pp. 611–615.

[18] P. Angkititrakul, D. Kwak, S. Choi, J. Kim, A. PhucPhan, A. Sathyanarayana, J.H. Hansen, Getting start with UTDrive: Driver-behavior modeling and assessment of distraction for in-vehicle speech systems, in: Eighth Annual Conference of the International Speech Communication Association, 2007.

[19] C. Craye, A. Rashwan, M.S. Kamel, F. Karray, A multi-modal driver fatigue and distraction assessment system, Int. J. Intell. Transp. Syst. Res. 14 (3) (2016) 173–194.

[20] N. Li, C. Busso, Predicting perceived visual and cognitive distractions of drivers with multimodal features, IEEE Trans. Intell. Transp. Syst. 16 (1) (2014) 51–65.

[21] R.A. Berri, A.G. Silva, R.S. Parpinelli, E. Girardi, R. Arthur, A pattern recognition system for detecting use of mobile phones while driving, in: International Conference on Computer Vision Theory and Applications, VISAPP, IEEE, 2014, pp. 411–418.

[22] T. Hoang Ngan Le, Y. Zheng, C. Zhu, K. Luu, M. Savvides, Multiple scale faster-RCNN approach to driver's cell-phone usage and hands on steering wheel detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2016, pp. 46–53.

[23] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards real-time object detection with region proposal networks, in: Advances in Neural Information Processing Systems, 2015, pp. 91–99.

[24] E. Ohn-Bar, S. Martin, A. Tawari, M.M. Trivedi, Head, eye, and hand patterns for driver activity recognition, in: 22nd International Conference on Pattern Recognition, IEEE, 2014, pp. 660–665.

[25] C.H. Zhao, B.L. Zhang, X.Z. Zhang, S.Q. Zhao, H.X. Li, Recognition of driving postures by combined features and random subspace ensemble of multilayer perceptron classifiers, Neural Comput. Appl. 22 (1) (2013) 175–184.
[26] C. Zhao, B. Zhang, J. He, J. Lian, Recognition of driving postures by contourlet transform and random forests, IET Intell. Transp. Syst. 6 (2) (2012) 161–168.
[27] C. Zhao, Y. Gao, J. He, J. Lian, Recognition of driving postures by multi-wavelet transform and multilayer perceptron classifier, Eng. Appl. Artif. Intell. 25 (8) (2012) 1677–1686.
[28] C. Zhao, B. Zhang, J. Lian, J. He, T. Lin, X. Zhang, Classification of driving postures by support vector machines, in: Sixth International Conference on Image and Graphics, IEEE, 2011, pp. 926–930.
[29] C. Yan, F. Coenen, B. Zhang, Driving posture recognition by convolutional neural networks, IET Comput. Vis. 10 (2) (2016) 103–114.
[30] Y. Xing, C. Lv, H. Wang, D. Cao, E. Velenis, F.-Y. Wang, Driver activity recognition for intelligent vehicles: A deep learning approach, IEEE Trans. Veh. Technol. 68 (6) (2019) 5379–5390.
[31] Y.-m. Song, S. Noh, J. Yu, C.-w. Park, B.-g. Lee, Background subtraction based on Gaussian mixture models using color and depth information, in: International Conference on Control, Automation and Information Sciences, ICCAIS 2014, IEEE, 2014, pp. 132–135.
[32] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.
[33] I. Jegham, A.B. Khalifa, I. Alouani, M.A. Mahjoub, Safe driving: Driver action recognition using SURF keypoints, in: 30th International Conference on Microelectronics, ICM, IEEE, 2018, pp. 60–63.
[34] O.D. Okon, L. Meng, Detecting distracted driving with deep learning, in: International Conference on Interactive Collaborative Robotics, Springer, 2017, pp. 170–179.
[35] StateFarm, State farm distracted driver detection dataset, 2016. https://www.kaggle.com/c/state-farm-distracted-driver-detection.
[36] H. Bay, T. Tuytelaars, L. Van Gool, Surf: Speeded up robust features, in: European Conference on Computer Vision, Springer, 2006, pp. 404–417.
[37] H.M. Eraqi, Y. Abouelnaga, M.H. Saad, M.N. Moustafa, Driver distraction identification with an ensemble of convolutional neural networks, J. Adv. Transp. 2019 (2019).
[38] B. Baheti, S. Gajre, S. Talbar, Detection of distracted driver using convolutional neural network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2018, pp. 1032–1038.
[39] H. Eraqi, Y. Abouelnaga, M. Saad, M. Moustafa, Distracted driver dataset, 2018. https://heshameraqi.github.io/distraction_detection.
[40] A. Behera, A. Keidel, B. Debnath, Context-driven multi-stream LSTM (M-LSTM) for recognizing fine-grained activity of drivers, in: German Conference on Pattern Recognition, Springer, 2018, pp. 298–314.
[41] A. Sharif Razavian, H. Azizpour, J. Sullivan, S. Carlsson, CNN features off-the-shelf: an astounding baseline for recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2014, pp. 806–813.
[42] K.B. Ahmed, B. Goel, P. Bharti, S. Chellappan, M. Bouhorma, Leveraging smartphone sensors to detect distracted driving activities, IEEE Trans. Intell. Transp. Syst. (2018).
[43] S. Tudor, S. Carey, R. Dubey, Development and evaluation of a dynamic virtual reality driving simulator, in: Proceedings of the 8th ACM International Conference on Pervasive Technologies Related to Assistive Environments, ACM, 2015, p. 55.
[44] S. Jafarnejad, G. Castignani, T. Engel, Non-intrusive distracted driving detection based on driving sensing data, in: Proceedings of the 4th International Conference on Vehicle Technology and Intelligent Transport Systems, VEHITS, 2018, pp. 178–186.
[45] A. Wesley, D. Shastri, I. Pavlidis, A novel method to monitor driver's distractions, in: CHI'10 Extended Abstracts on Human Factors in Computing Systems, ACM, 2010, pp. 4273–4278.
[46] C. Bo, X. Jian, X.-Y. Li, X. Mao, Y. Wang, F. Li, You're driving and texting: detecting drivers using personal smart phones by leveraging inertial sensors, in: Proceedings of the 19th Annual International Conference on Mobile Computing & Networking, ACM, 2013, pp. 199–202.
[47] H.A. Shabeer, R. Wahidabanu, Averting mobile phone use while driving and technique to locate the mobile phone used vehicle, Procedia Eng. 30 (2012) 623–630.
[48] P. Singh, N. Juneja, S. Kapoor, Using mobile phone sensors to detect driving behavior, in: Proceedings of the 3rd ACM Symposium on Computing for Development, ACM, 2013, p. 53.
[49] M. Fazeen, B. Gozick, R. Dantu, M. Bhukhiya, M.C. González, Safe driving using mobile phones, IEEE Trans. Intell. Transp. Syst. 13 (3) (2012) 1462–1468.
[50] D. Dörr, D. Grabengiesser, F. Gauterin, Online driving style recognition using fuzzy logic, in: 17th International IEEE Conference on Intelligent Transportation Systems, ITSC, IEEE, 2014, pp. 1021–1026.
[51] V. Vaitkus, P. Lengvenis, G. Žylius, Driving style classification using long-term accelerometer information, in: 19th International Conference on Methods and Models in Automation and Robotics, MMAR, IEEE, 2014, pp. 641–644.
[52] D.A. Johnson, M.M. Trivedi, Driving style recognition using a smartphone as a sensor platform, in: 14th International IEEE Conference on Intelligent Transportation Systems, ITSC, IEEE, 2011, pp. 1609–1615.
[53] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: Proceedings of the 32nd International Conference on Machine Learning, ICML, 2015, pp. 448–456.
[54] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[55] S.J. Pan, Q. Yang, A survey on transfer learning, IEEE Trans. Knowl. Data Eng. 22 (10) (2009) 1345–1359.
[56] L. Torrey, J. Shavlik, Transfer learning, in: Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques, IGI Global, 2010, pp. 242–264.
[57] ILSVRC, ImageNet large scale visual recognition challenges, 2010-2017, http://image-net.org/. (Accessed 28 September 2019).
[58] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2016, pp. 2818–2826.
[59] Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu, M. Song, Neural style transfer: A review, IEEE Trans. Vis. Comput. Graphics (2019).
[60] S. Taheri, Ö. Toygar, On the use of DAG-CNN architecture for age estimation with multi-stage features fusion, Neurocomputing 329 (2019) 300–310.
[61] X.-S. Wei, C.-W. Xie, J. Wu, C. Shen, Mask-CNN: Localizing parts and selecting descriptors for fine-grained bird species categorization, Pattern Recognit. 76 (2018) 704–714.
[62] N. Gozuacik, C.O. Sakar, Turkish movie genre classification from poster images using convolutional neural networks, in: 2019 11th International Conference on Electrical and Electronics Engineering, ELECO, IEEE, 2019, pp. 930–934.
[63] A.R. Saikia, K. Bora, L.B. Mahanta, A.K. Das, Comparative assessment of CNN architectures for classification of breast FNAC images, Tissue Cell 57 (2019) 8–14.
[64] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Proceedings of the 3rd International Conference on Learning Representations, ICLR, 2014.
[65] Y. Bengio, P. Simard, P. Frasconi, et al., Learning long-term dependencies with gradient descent is difficult, IEEE Trans. Neural Netw. 5 (2) (1994) 157–166.
[66] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[67] M.W. Lam, X. Chen, S. Hu, J. Yu, X. Liu, H. Meng, Gaussian process LSTM recurrent neural network language models for speech recognition, in: IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, IEEE, 2019, pp. 7235–7239.
[68] J. Lorenzo-Trueba, G.E. Henter, S. Takaki, J. Yamagishi, Y. Morino, Y. Ochiai, Investigating different representations for modeling and controlling multiple emotions in DNN-based speech synthesis, Speech Commun. 99 (2018) 135–143.
[69] J. Xu, F. Huang, X. Zhang, S. Wang, C. Li, Z. Li, Y. He, Sentiment analysis of social images via hierarchical deep fusion of content and links, Appl. Soft Comput. 80 (2019) 387–399.
[70] C.O. Sakar, S.O. Polat, M. Katircioglu, Y. Kastro, Real-time prediction of online shoppers' purchasing intention using multilayer perceptron and LSTM recurrent neural networks, Neural Comput. Appl. 31 (10) (2019) 6893–6908.
[71] Y. Liu, X. Zhang, F. Huang, X. Tang, Z. Li, Visual question answering via Attention-based syntactic structure tree-LSTM, Appl. Soft Comput. 82 (2019) 105584.
[72] R. Ghosh, C. Vamshi, P. Kumar, RNN based online handwritten word recognition in Devanagari and Bengali scripts using horizontal zoning, Pattern Recognit. 92 (2019) 203–218.
[73] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: Synthetic minority over-sampling technique, J. Artificial Intelligence Res. 16 (2002) 321–357.