Estimating the Driver Status Using Long Short Term Memory

Shokoufeh Monjezi Kouchak([✉]) and Ashraf Gaffar

Arizona State University, Tempe, AZ, USA
smonjezi@asu.edu

Abstract. Driver distraction is one of the leading causes of fatal car accidents. Driver distraction is any task that diverts the driver attention from the primary task of driving and increases the driver's cognitive load. Detecting potentially dangerous driving situations or automating some repetitive tasks, using Advanced Driver Assistance Systems (ADAS), and using autonomous vehicles to reduce human errors while driving are two suggested solutions to diminish driver distraction. These solutions have some advantages, but they suffer from their inherent inability to detect all potentially dangerous driving situations. Besides, autonomous vehicles and ADAS depend on sensors. As a result, their accuracy diminishes significantly in adverse conditions. Analyzing driver behavior using machine learning methods and estimating the distraction level of drivers can be used to detect potentially hazardous situations and warn the drivers. We conducted an experiment in eight different driving scenarios and collected a large dataset from driving data and driver related data. We chose Long Short Term Memory (LSTM) as our machine learning method. We built and trained a stacked LSTM network to estimate the driver status using a sequence of driving data vectors. Each driving data vector has 10 driving related features. We can accurately estimate the driver status with no external devices and only using cars Can-Bus data.

Keywords: Recurrent Neural Network · Driver distraction · Deep learning · Long Short Term Memory Network

1 Introduction

Everyday approximately nine people die and more than 1,000 are injured in car crashes that are caused by distracted drivers [1]. More than 90% of car accidents happen due to human error [2]. Using a cell phone or interacting with the car infotainment system significantly increases drivers' cognitive load and causes driver distraction. Car crash is the leading cause of teenage death [3]. [4] Mentions that 21% of teenagers involved in fatal car accidents were distracted by their cellphone. Teenagers are four times more likely to have car crashes while they are texting or talking on the phone [4]. Being an attentive driver can reduce the chance of car accidents significantly, but it is not the ultimate solution for driver distraction since people do distractive tasks such as texting although they know it is a serious issue for their safety and can lead to a car crash. AAA Traffic Safety 2016 report mentions that from 2,501 drivers that participated in their research almost 90% said texting while driving is very dangerous, but 18% of them admitted to texting while driving in the past month [5].

Driver distraction is any task that diverts the driver attention from the primary task of driving and increases the cognitive load of the driver [6, 7]. There are four types of driver distraction including visual, manual, audio and cognitive distraction. Some distracting tasks such as texting cause a combination of these types of distraction, so they are more dangerous compared to tasks that only cause one type of distraction [8, 9]. For instance, texting causes manual, visual and cognitive distraction and it makes texting more dangerous than drunk driving. Based on NHTSA research texting while driving is six times more dangerous than driving intoxicated [10].

Enhancing the design of user interface in cars [11, 12] to make it more suitable for cars' environment can reduce the cognitive load of interacting with the car's info-tainment system. Autonomous vehicles and Advanced Driver Assistant Systems (ADAS) are two suggested solutions for enhancing driver safety [13–16]. ADAS can handle some specific dangerous situations but they can't detect all potentially menacing situations [17]. Besides, although autonomous vehicles can drive automatically in normal situations using complex sensors and sophisticated artificial intelligence methods, they can't handle some unexpected and extreme events. The human driver needs to be ready for taking control of the car in emergency situations [18, 19].

Driver behavior analysis methods have been used in [20–22] to detect the driver's abnormal behavior or the level of the driver's cognitive load to estimate and reduce driver distraction. A variety of machine learning methods such as Neural Network, Hidden Markov Model and Support Vector Machine have been used to analyze driver behavior using driver status data, car related data or combination of them [22]. Deep learning methods such as convolutional neural networks, deep neural networks and recurrent neural networks outperform traditional machine learning approaches in many safety-related applications such as pedestrian detection [13]. Driving is an intricated task, so deep learning methods are suitable choices to extract and learn complex patterns of driving.

Driving is a continuous task and the driving situation in each time step depends on several previous steps and influences several next steps. Markov Model is a machine learning approach that has memory making it a suitable choice for applications with dependent inputs. It has been used in many driving safety applications successfully [23, 24]. In Hidden Markov Model all possible actions and states need to be defined in advance, so it works accurately when we want to detect a specific condition or analyze few specific maneuvers. On the other hand, if we have a large number of states and possible actions or if all states can't be defined in advance Markov Model is not a suitable choice for our system. In these cases, Recurrent Neural Networks (RNN) can be used as a more appropriate method that can learn intricated patterns with no need to have previous information about the model [25–27]. In this paper, we use a Long Short Term Memory (LSTM) network which is a type of Recurrent Neural Network to estimate the driver behavior using both scaled and not scaled driving data. We collected a large dataset of driving and driver behavior data vectors by conducting an experiment in 8 different driving scenarios and used the dataset to train an LSTM network which estimates the driver behavior using driving data. In Sect. 2, we talk about RNN and related works. Section 3 is the experiment and Sect. 4 is data. In Sect. 5, we discuss the model that we built using an LSTM network. Section 6 is results, and Sect. 7 is the conclusion.

2 Recurrent Neural Network

Traditional Neural Networks assume that all input samples are independent of each other, so after training the network using each sample, all information about the sample removes and the next sample doesn't use any information from the previous ones. Besides, they use fixed size input and output data. These assumptions are not true in some real-life applications such as speech recognition, language translation, and autonomous driving. People consider their previous experiments and knowledge in each time to make a decision and unlike traditional neural network for many real-life applications, the feature vector in each moment depends on one or several previous samples. Moreover, in some applications such as language translation, we need to deal with variable length inputs and outputs [25].

A recurrent neural network (RNN) is a type of neural network that solved these shortages of traditional neural networks using a loop in the network that allows information to persist. A recurrent neural network can be considered as several copies of the same network that each network passes a message to a successor. The chain shape of unrolled RNN shows that they are a specific architecture of neural networks to use for learning a sequence of dependent data. In this type of network, each output influenced not only by the current input of the network but also all inputs that have been fed to the network until the current step. RNN networks outperform many machine learning methods in real-life applications such as speech recognition and language translation [26, 27]. RNNs have been used in many car-related applications such as autonomous driving, driver behavior analysis and driving safety. In these car applications, their performance was much better than other machine learning approaches that don't have memory.

Various psychological conditions like sleepiness, fatigue, and distraction have an adverse effect on driver performance and can lead to fatal car accidents. [28] Discussed a model that detects driver potentially dangerous psychological conditions such as fatigue using a brain-computer interface. It proposed a new recurrent neural network architecture called Recurrent Self-evolving Fuzzy Neural Network. This model finds the correlation between the driver's brain activity, that monitors using EEG, and the driver's fatigue level [28].

[29] Proposed a data collection and data analysis framework called "DarNet". It can detect and classify driver behaviors. The framework has two parts including a data collection system and data analyzing part. Images that are collected by a face camera and Internal Measurement Unit (IMU) data from a mobile device are the inputs of this framework. They used deep learning methods including Long Short Term Memory networks and Convolutional neural network to classify driver behavior and reached 87.02% accuracy.

In this paper, we discuss a model that predicts the driver status using a sequence of driving data vectors and a stacked Long Short Term Memory (LSTM) network. We only used cars Can-Bus data and we didn't use any external devices such as camera or external sensors to make our dataset. We try both scaled and not-scaled data then we compared the results of them. Driver status in this paper shows if the driver is interacting with car infotainment system or not. Besides, if the driver has interaction with car infotainment system, what are the features of this interaction. We defined four features for each interaction including the number of errors while interaction, the length of interaction or response time, the number of navigation steps that the driver needs to pass in order to complete the interaction and the driving mode.

3 Experiment

We conducted our experiment in the HCaI lab using a Drive Safety Research simulator DS-600 which is fully integrated driving simulation system that includes a minimum 180° wraparound display, multi-channel audio/visual system, full-width automobile cab (Ford Focus) including windshield, center console, driver and passenger seats, dash and instrumentation and real-time vehicle motion simulation. This simulator provides a variety of road types such as urban road and highway. Besides, the different driving mode such as night, rain, fog and snow can be chosen for each road. We designed an urban road with high traffic. Figure 1 shows the designed road.

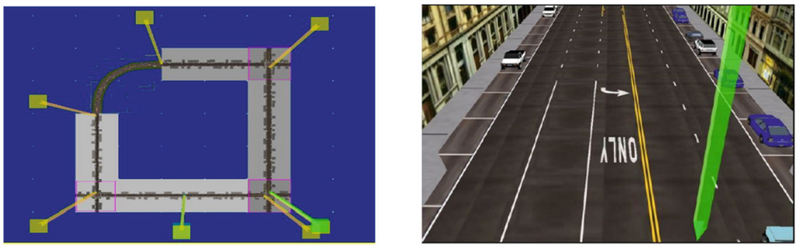

Fig. 1. Designed road

Four driving modes have been defined including Day, Night, Fog and Fog & night. The day mode is called the ideal mode, the Night mode and the Fog mode are adverse modes and the Fog & Night mode is the double adverse mode. We defined eight driving scenarios including four distracted and four non-distracted ones, so in each driving mode, we have distracted and non-distracted driving scenarios. In a distracted scenario, the driver interacts with the car interface continuously. An android application has been used as the car interface in this experiment. The application was hosted on the Android v4.4.2 based Samsung Galaxy Tab4 8.0 which was connected to the HyperDrive simulator. Figure 2(a) shows the place that we put the tablet on it to simulate the screen of car infotainment system. This application simulates car infotainment system interface in modern cars. We installed the tablet in the middle console next to the driver and ran an android application designed for the experiment. The main page of the application shows the main screen of car infotainment system and the driver can navigate in this application like car infotainment system.

In this experiment we wanted to detect if the driver had any interaction with our application. In non-distracted modes, the driver only focuses on the primary task of driving and we use the collected data in the non-distracted modes as the baseline of the model since they show no interaction. On the other hand, if the driver interacts with car infotainment system, we want to detect the features of this interaction since all interactions are not equally distracting.

We used the minimalist design discussed in [12] as the navigation model of the interface (Fig. 2(b)). In this navigation model, each command can be reached by 2, 3 or 4 steps of navigation, so we defined three types of tasks including 2-step, 3-step and 4-step tasks and use them to distract drivers in distracted scenarios. In distracted scenarios, we asked drivers to do some tasks on the interface considering this fact that driving is the primary task, and he/she shouldn't only focus on the requested task that has less priority compared to the primary task of driving. We define the task as reaching to a specific application on the interface that based on our interface design it needs 2 or 3 or 4 steps of navigation.

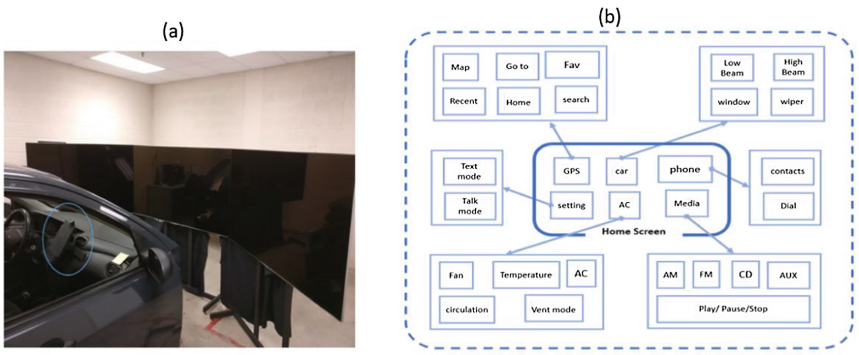

Fig. 2. (a) The blue circle shows the place that we put the tablet in the simulator (b) The navigation model that we used in our infotainment system (Color figure online)

We invited 35 volunteers to participate in this experiment by taking around 45 min of simulated drive. They were undergraduate and graduate students of Arizona State University in range 20 to 35 years old and they had at least two years of driving experience. Each volunteer was trained for 10 min before starting the experiment to become familiar with the simulator and the car's interface. After that, they drove in eight different scenarios. In non-distracted modes, they were asked to focus on the primary task of driving. In distracted scenarios, we chose some tasks from each type of tasks (2-step, 3-step and 4-step), and we tried to have equal number of tasks from each type. After that, we asked the driver to start the trip. After few seconds we started asking the driver to do some tasks on the interface, reach specific application on the interface, and we put the same gap between each two tasks since we want to have a border between the driving data related to each task. They were asked to give the highest priority to the primary task of driving and drive as realistic as possible. Besides, we added some events, such as pedestrians and bike drivers that jump into the road, randomly to each scenario to reduce the learning effect. We observed the driver behavior during each task and collected 4 data four each task:

1. The number of errors during a task: an error is defined as touch a wrong icon on the interface, not following traffic rules or have an accident.
2. Response time: the response time shows the length of interaction. We consider the length of interaction from the moment that we asked the driver to do the task and the moment that the task is done.
3. The number of navigation steps. We have three types of tasks including the 2-step, 3-step and 4-step task. The number of navigation steps is related to the task difficulty level since more navigation steps and longer tasks need more attention and cause higher level of distraction.
4. The driving mode which is the name of driving scenario. We use number 1 to 4 to shows non-distracted scenarios and 5 to 8 to display distracted scenarios.

4 Data

In each trip, on average 19000 data vectors have been collected by the simulator and each of them has 53 features, so we collected 5.3 million driving data vectors during this experiment. For each task that we asked the driver to perform, we collected four features including the mode of driving, the number of navigation steps, the driver response time and the number of errors. In all, we collected 2025 driver related data manually in this experiment. The sampling rate of the simulator was set on 60 samples per second which was the maximum possible rate. We wanted to collect as much data as possible and save it as a master data to be used for this and future experiments, so we decided to use this high sampling rate since it did not have any negative effect on our experiment or our budget. But we didn't use all samples in our model since in our application using 60 samples per second of driving is computationally very expensive for our machine. Therefore, we decided to compress the dataset. We replaced each 20 data vectors with their mean value. Future experiments will investigate if different methods of compressing this sampling rate (like median, mode, or any other function) would have an effect on the system training time or intelligence level.

For each driver feature vectors that we collected there are a large number of correlated driving data vectors that were collected by the simulator. The number of these vectors depends on the driver response time. To find the corresponding driving related vectors to each driver data vector we divided the collected driving vectors in each trip based on the length of the response time of all tasks that have been done during the trip. We calculated the sum of the response time for volunteer X in mode Y and calculate the portion of driving data vectors samples which are correlated to each driver data vector (1). In this equation, n shows the number of collected driver data vectors in a specific trip and the result shows the percentage of collected driving data vectors which is related to specific driver data vectors.

$$Percentage(i) = [response(i)/\sum_{k=0}^{n} response(k)] * 100 \tag{1}$$

The driving data vectors have 53 features, and we chose 10 of them including speed limit, brake, velocity, steering wheel, longitude accelerating, lateral accelerating, lane position, accelerating, headway time and headway distance as inputs of our model. We used a paired t-test between distracted and non-distracted scenarios in each mode to detect the features that are significantly different between distracted and non-distracted modes (Table 1) and the combination of these 10 features shows significantly different between our scenarios.

Table 1. T-test results for distracted and non-distracted scenarios

	Day distracted vs. day non-distracted	Night distracted vs. night non-distracted	Fog distracted vs. fog non-distracted	Fog night distracted vs. fog night non-distracted
Velocity	Not significant	Extremely significant	Significant	Very significant
Lane position	Extremely significant	Extremely significant	Extremely significant	Extremely significant
Steering	Extremely significant	Very significant	Not significant	Extremely significant
Speed limit	Not significant	Not significant	Significant	Not significant
Accelerating	Extremely significant	Extremely significant	Extremely significant	Extremely significant
Brake	Very significant	Not significant	Not significant	Extremely significant
Longitude accelerating	Not significant	Extremely significant	Not significant	Not significant
Lateral accelerating	Not significant	Not significant	Not significant	Not significant
Headway time	Extremely significant	Not significant	Significant	Not significant
Headway distance	Extremely significant	Extremely significant	Extremely significant	Not significant

5 LSTM Model

We chose a multi-input single-output LSTM network for our model. The input of the network is a sequence of driving data and the output is a single driver data vector. We built three multi-layers LSTM networks with 2, 3 and 4 LSTM layers using Keras library. We put 50 neurons in each LSTM layer to check a different combination of batch sizes, learning rates and activation functions for this model. Finally, we chose 100 as the batch size, 0.0001 as the model's learning rate and ReLU as the activation function. We tried both scaled and not scaled data for training this network. We chose 80% of the dataset as training data and 20% as testing data. Besides, we used 20% of the training data as validating data during training process.

We trained the two-layer network with different numbers of LSTM neurons from 10 to 1000 and finally, we choose three numbers including 50, 150 and 300 as the number of LSTM neurons in each LSTM layer and analyze the effects of a low, medium and large number of LSTM neurons on the final result. Less than 50 neurons resulted in underfitting and increasing the number of LSTM neurons from 300 to 1000 didn't improve the accuracy of the network and only caused overfitting. After 300 LSTM neurons instead of increasing the number of neurons, we tried deeper networks to improve the accuracy of the model. After training fully connected LSTM networks, we tried 20%, 40%, and 50% dropout to reduce overfitting chance of the model but using drop-out didn't have a positive effect on the final accuracy.

6 Results

We tried pure data first and trained three different LSTM networks with two, three and four LSTM layers. The learning rate is 0.0001, the batch size is 100 and the activation function is ReLU for these networks. Table 2 shows the train, test and validation error of them using 50, 150 and 300 as the number of neurons in LSTM layers. The layer column shows the number of LSTM layers, the LSTM column shows the number of neurons in each LSTM layer, MAE is Mean Absolute Error and MSE is Mean Square Error of the model.

Table 2. Summary of multi-layers LSTM networks results with not-scaled data

Layer	LSTM	Train MAE	Train MSE	Val MAE	Val MSE	Test MAE	Test MSE
2-no scale	50	0.95	2.35	1.4	6.51	1.38	7.25
2-no scale	150	0.54	0.78	1.49	6.01	1.41	6.37
2-no scale	300	0.39	0.56	1.45	5.46	1.51	7.33
3-no scale	50	0.76	1.33	1.44	6.51	1.38	7.25
3-no scale	150	0.33	0.47	1.39	5.57	1.48	6.93
3-no scale	300	0.31	0.46	1.4	6.09	1.34	5.33
4-no scale	50	0.76	1.38	1.38	5.6	1.33	5.93
4-no scale	150	0.4	0.5	1.4	5.99	1.46	7.19
4-no scale	300	0.17	0.24	1.49	7.68	1.39	5.92

For the two-layer networks, using 50 neurons in LSTM layers resulted in the most accurate model since it has the minimum validation and test error. Besides, the gap between the train and test error is less than the rest of them. In the three-layer networks, although the network with 300 LSTM neurons has the minimum test error the performance of the network with 50 LSTM neurons is better since it has the minimum gap between the train and test error that shows the model trained well and it is not overfitted. Besides, the test error of this network is close to the network with 300 LSTM neurons. In the four-layer network, the model with 50 LSTM neurons resulted in the minimum validation and test error. Moreover, it has the minimum gap between the train and the test error, so it's trained better than the rest of the four-layer networks.

In sum, all three networks have their best performance using 50 neurons in their LSTM layers. Although adding more neurons enhanced their train error it didn't have a positive effect on the test error and the networks went toward overfitting. We scaled all input and output data then trained all networks again using scaled data. Table 3 shows the summary of three different networks with scaled data and 50,150 and 300 as the number of LSTM layer's neuron. In the two-layer LSTM network, we can see overfitting in all networks that we trained and the minimum gap between train and test error is 1. When we increased the number of neurons from 50 to 300 the test error didn't change much but overfitting decreased. For the three-layer network, the network with 50 LSTM neurons shows the best performance and the least overfitting but the performance of two other networks is similar to the two-layer networks with the same number of neurons. Four-layer network has better performance and the least accuracy for all three number of neurons that we tried.

Table 3. Summary of multi-layers LSTM networks results with scaled data

Layer	LSTM	Train MAE	Train MSE	Val MAE	Val MSE	Test MAE	Test MSE
2-scaled	50	0.11	0.03	1.01	3.36	1	2.67
2-scaled	150	0.22	0.1	0.99	2.68	1.04	3.36
2-scaled	300	0.36	0.24	0.98	2.58	1.03	3.39
3-scaled	50	0.4	0.32	1.01	3.33	0.98	2.88
3-scaled	150	0.15	0.05	1.03	3.42	1.03	3
3-scaled	300	0.12	0.03	0.87	2.1	1	3.4
4-scaled	50	0.57	0.61	0.97	3.36	0.9	2.19
4-scaled	150	0.5	0.48	1	3.6	0.93	2.55
4-scaled	300	0.3	0.19	0.9	2.9	1.05	2.97

In sum, the four-layer network with 50 neurons resulted in the minimum test error which is 0.9 and it has the least gap between the train and the test error, so we can say that the network trained well and doesn't have overfitting. We tried more LSTM layers, but the final result was less accurate than the four-layer network. Increasing the number of neurons in shallower networks enhance the accuracy and in deeper networks, it just increased the overfitting chance.

7 Conclusion

Driver distraction is any task that diverts the driver attention away from the primary task of driving. Advances Driver Assistant Systems, Autonomous Vehicles and driver behavior analyzing are some suggested solutions to reduce driver distraction. Driver car infotainment system interaction is one of the main sources of driver distraction. Although interacting with car infotainment system, even for short time, causes distraction, the level of distraction and cognitive load which is caused by each task depends on many different factors such as the length of distraction, the context of driving and the complexity of the task. If the driver can do a specific task on the car infotainment system in a short time and find a specific application with few simple navigation steps, the distraction level that is caused by this task is much less than a long task such as texting or navigating in a complex interface to reach an application.

We defined four features for each interaction including number of errors, response time, number of navigation steps and the mode of driving. We used 10 driving data and a stacked LSTM network to detect the driver status that is defined by these four features. We reached 0.95 train MAE and 1.38 test MAE with not scaled data, two-layer LSTM network and 50 neurons in each LSTM layer. We trained the network with scaled data and reached 0.57 Train MAE and 0.9 test MAE with four-layer LSTM network and 50 neurons in each LSTM layer. In sum, we detect if the driver is distracted by interacting with car infotainment system or not and if he/she is interacting with the car infotainment system, what are the features of this interaction. As future work, we can extend this experiment using more driving scenario and new distracting task that cover a larger subset of the possible distracting task while driving. Besides, we can use the output of the model as the input of a decision system and give each feature a specific weight to calculate the driver distraction level more accurately.

Acknowledgement. We acknowledge the effort and participation of all ASU undergraduate and graduate students who enthusiastically participated in data collection and analysis of some of these experiments as part of their course load or as volunteers.

References

1. National Center for Statistics and Analysis, Distracted Driving: 2015, in Traffic Safety Research Notes. DOT HS 812 381. National Highway Traffic Safety Administration, Washington, D.C., March 2017
2. Injury Facts, Motor Vehicles Safety Issues. https://injuryfacts.nsc.org/motor-vehicle/motor-vehicle-safety-issues/. Accessed 20 Oct 2018
3. Ferguson, R., Green, A., Blau, E., Walker, L.: Teens in Cars. Safe Kids Worldwide, Washington, DC, May 2014
4. National Safety Council: Teens' Biggest Safety Threat is Sitting on the Driveway. https://www.nsc.org/road-safety/safety-topics
5. AAA Foundation For Traffic Safety, 2016 Traffic Safety Culture Index, February 2017. https://aaafoundation.org/wp-content/uploads/2017/11/2016TrafficSafetyCultureIndexReport.pdf. Accessed 10 Aug 2018
6. Gaffar, A., Monjezi Kouchak, S.: Using artificial intelligence to automatically customize modern car infotainment systems. In: Proceedings on the International Conference on Artificial Intelligence (ICAI), pp. 151–156 (2016)
7. National Center for Statistics and Analysis: Driver electronic device use in 2017, Traffic Safety Facts Research Note. Report No. DOT HS 812 665. National Highway Traffic Safety Administration, Washington DC, January 2019
8. Gaffar, A., Monjezi Kouchak, S.: Using simplified grammar for voice commands to decrease driver distraction. In: The 14th International Conference on Embedded System, pp. 23–28 (2016)
9. Vegega, M., Jones, B., Monk, C.: Understanding the effects of distracted driving and developing strategies to reduce resulting deaths and injuries: a report to congress. Report No. DOT HS 812 053. National Highway Traffic Safety Administration, Washington, DC, December 2013
10. Todd, W.: It is Time for a 'Parental Control, No Texting While Driving' Phone. Forbes Business, 18 September 2012

11. Gaffar, A., Monjezi Kouchak, S.: Quantitative driving safety assessment using interaction design benchmarking. In: IEEE Advanced and Trusted Computing (ATC 2017), San Francisco Bay Area, USA, 4–8 August 2017 (2017)
12. Gaffar, A., Monjezi Kouchak, S.: Minimalist design: an optimized solution for intelligent interactive infotainment systems. In: IEEE IntelliSys, the International Conference on Intelligent Systems and Artificial Intelligence, London, 7–8 September 2017
13. Campbell, M., Egerstedt, M., How, J., Murray, R.: Autonomous driving in urban environments: approaches, lessons and challenges. Philos. Trans. R. Soc. **368**(1928), 4649–4672 (2010). https://doi.org/10.1098/rsta.2010.0110
14. Gaffar, A., Monjezi Kouchak, S.: Undesign: future consideration on end-of-life of driver cars. In: IEEE Advanced and Trusted Computing (ATC 2017), San Francisco Bay Area, USA, 4–8 August 2017 (2017)
15. Lu, M., Werers, K., Heijden, R.: Technical feasibility of advanced driver assistance systems (ADAS) for road traffic safety. Transp. Plann. Technol. **28**(3), 167–187 (2005). https://doi.org/10.1080/03081060500120282
16. Monjezi Kouchak, S., Gaffar, A.: Determinism in future cars: why autonomous trucks are easier to design. In: IEEE Advanced and Trusted Computing (ATC 2017), San Francisco Bay Area, USA, 4–8 August 2017 (2017). https://doi.org/10.1109/uic-atc.2017.8397598
17. Brooks, C., Rakotonirainy, A.: In-vehicle technologies, advanced driver assistance systems and driver distraction: research challenges. In: International Conference on Driver Distraction, Sydney, Australia, 2–3 June 2005 (2005)
18. Finn, A., Scheding, S.: Developments and Challenges for Autonomous Unmanned Vehicles. Intelligent Systems Reference Library, vol. 3. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-10704-7. ISBN: 978-3-642-10703-0
19. Barabás, I., Todoruţ, A., Cordoş, N., Molea, A.: Current challenges in autonomous driving. IOP Conf. Ser.: Mater. Sci. Eng. **252**, 012096 (2017)
20. Monjezi Kouchak, S., Gaffar, A.: Non-intrusive distraction pattern detection using behavior triangulation method. In: International Conference on Computational Science and Computational Intelligence (CSCI), 14–16 December 2017. https://doi.org/10.1109/csci.2017.140
21. Goodrich, M., Quigley, M.: Learning haptic feedback for guiding driver behavior. In: IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583) (2004)
22. Meiring, G., Myburgh, H.: A review of intelligent driving style analysis systems and related artificial intelligence algorithms. Sensors **15**(12), 30653–30682 (2015)
23. Gindele, T., Brechtel, S., Dillmann, R.: A probabilistic model for estimating driver behaviors and vehicle trajectories in traffic environments. In: 13th International IEEE Conference on Intelligent Transportation Systems (2010)
24. Hou, H., Jin, L., Niu, Q., Sun, Y., Lu, M.: Driver intention recognition method using continuous hidden Markov model. Int. J. Comput. Intell. Syst. **4**(3), 386–393 (2011)
25. Norvig, P., Russell, S.: Artificial Intelligence: A Modern Approach, 3rd edn. Pearson, London (2014). ISBN 0136042597
26. Gibson, A., Patterson, J.: Deep Learning. Oreilly, Sebastopol (2017). ISBN 978-1491914250
27. Mandic, D., Chambers, J.: Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability. Wiley, Hoboken (2010). ISBN 0471495174
28. Liu, Y., Lin, Y., Wu, S., Chuang, C., Lin, C.: Brain dynamics in predicting driving fatigue using a recurrent self-evolving fuzzy neural network. IEEE Trans. Neural Netw. Learn. Syst. **27**, 347–359 (2017)
29. Streiffer, C., Raghavendra, R., Benson, T., Srivatsa, M.: Darnet: a deep learning solution for distracted driving detection. In: The 18th ACM/IFIP/USENIX Middleware Conference (2017). https://doi.org/10.1145/3154448.3154452