Application of Machine Learning and Image Recognition for Driver Attention Monitoring

Manav Tailor, Jahangir Ali, Xinrui Yu, Won-Jae Yi, and Jafar Saniie
Embedded Computing and Signal Processing (ECASP) Research Laboratory (http://ecasp.ece.iit.edu)
Department of Electrical and Computer Engineering
Illinois Institute of Technology, Chicago IL, U.S.A.

***Abstract*—This paper presents a real-time prototype system for monitoring the distraction levels of the driver. Due to the nature of high traffic conditions commonly seen nowadays, accidents are highly likely to occur as drivers cannot always recognize their exhaustion levels themselves. We utilize a single low-cost camera facing the driver connected to a single-board computer; a series of frame captures from the camera are fed to a neural network, and a pattern detection algorithm to predict the driver's distraction level is utilized. All training is conducted under personalized training sets to increase accuracy and to match an individual's driving patterns as accurately as possible. This system is designed to serve as a baseline for further system development, and many vital sub-components can be changed regarding input data type and choices of machine learning algorithms.**

Keywords—Artificial Intelligence, Machine Learning, Driver Attention Monitoring, Edge Computing, Real-time Image Processing

I. Introduction

Many collisions on the road occur due to the driver being distracted or not in a proper state to drive. As stated in [1], "In the United States, over 3,100 people were killed, and about 424,000 were injured in crashes involving a distracted driver in 2019". Systems have been in place to assist in some of these scenarios, such as for drowsy drivers. Also as indicated in [1], "Some states have installed rumble strips on highways to alert drowsy, distracted, or otherwise inattentive drivers that they are veering off the road. These rumble strips are effective at reducing certain types of crashes". These rumble strips may serve as effective in certain situations and areas in the roadway, yet they are only there to passively prevent collisions, not actively. A few systems attempted to prevent drivers from getting distracted by items, people, and pets in their vehicles, such as chatting with passengers or phone usage. Companies have worked on reducing phone usage while driving, for example, Tesla's center console [2]. This console allows the driver and passenger to place their phone on charge and close the compartment so their phones are out-of-sight. However, this works on the honor system and allows the driver to look at their phone by choosing not to close the compartment.

Furthermore, not all cars or people have the luxury of these compartments in their vehicles. As this is just one scenario that was examined, many other situations have not been accounted for, and many solutions to these problems are based on the honor system. Creating a system that can monitor a driver using a singular camera would be beneficial in a vehicle. Not only would a system like such be integrated as an after-market product but it can also be designed to recognize the driver's driving patterns and have higher accuracy in judging the state of the driver. Creating a mechanical stimulation, such as a beep or a buzz, can alert the driver when they are not paying attention to the road and the surrounding environment. The long-term effect of such would be to subconsciously train the driver to drive with more focus on the road.

II. Related Work

The concept of monitoring the driver is an open-ended problem; many papers and projects exist on this topic.

A. Real-time monitoring of driver drowsiness on mobile platforms using 3D neural networks

In this paper, a 3D convolutional neural network was utilized to create a system for real-time video monitoring of the driver, with considerable attention given to model deployment on mobile phones and other cost-effective vehicle-based computers. This has reduced the need for specialized hardware, enabling a cost-effective solution [3]. Drowsiness was the driver state targeted for measurement and the drowsiness detection was broken down into three categories: vehicle-based measurements, physiological measurements, and computer vision techniques. In this study, computer vision was utilized.

B. NVIDIA DRIVE IX cabin perception software

NVIDIA DRIVE IX is an open and scalable cockpit software platform that provides a range of in-cabin experiences. These include functions such as intelligent visualization, augmented reality, virtual reality, conversational AI, and interior sensing. With these sensing features, the platform can utilize the Audio and Visual (AV) system to monitor and ensure the driver is paying attention to the road [4]. Facial expressions have many different meanings, such as a shift or wrinkle of the brow. DRIVE IX utilizes multiple DNNs (Deep Neural Networks) to decipher expressions. The first DNN detects the face; the second identifies reference markings and points. Additionally, various DNNs work to determine if the driver is paying attention. GazeNet DNN is designed to track gazes by creating a map of vectors corresponding to the driver's eye and the road. This map is then used to see if the driver can see hazards on the road. SleepNet monitors drowsiness by determining whether the driver's eyes are open or closed and can determine a level of exhaustion. ActivityNet monitors the drivers' activities, such as

phone usage, hands-on or off-the-wheel, and their attention to events outside the vehicle. Additionally, DRIVE IX monitors the drivers' sitting positions, emotions, and conversations via various methods.

C. *Real-Time Driver's Focus of Attention Extraction and Prediction using Deep Learning*

This paper discusses the design, implementation, and evaluation of DFaep, a deep learning network to examine, estimate, and predict drivers' focus of attention using dual low-cost dash cameras for driver-centric and car-centric views [5]. DFaep deployed real-time head movement tracking to monitor the driver's gaze. The driver's attention was then broken down into (11) grids within (5) gaze zones in the vehicle. A DNN then takes this data and determines whether the driver is distracted. DFaep showed an average accuracy of 99.38%.

D. *Existing Studies for Driver Drowsiness Detection utilizing Drowsiness Detection*

Table I shows some other studies on this topic that were not discussed in this paper.

TABLE I. LIST OF CURRENT STUDIES ON DRIVER ATTENTION MONITORING

Study	Method of Deployment
Jiménez et al. [6]	Haar classifiers for the head, eye, and mouth segment detection in video frames; a NN then classifies the level of driver distraction in each frame
Ying et al. [7]	Contour detection methods to locate the face, mouth, and eyes; a NN follows and assesses the state of each feature
Ribarić et al. [8]	Classifies head rotation, eye, and mouth openness to issue alarms based on drowsiness levels
Ji et al. [9]	Bayesian network is fed head and eyelid movements with facial features to access fatigue
Jiangwei et al. [10]	Single feature NN to classify whether the driver is dozing, talking, or silent; focuses on the mouth
Rong-ben et al. [11]	Single feature NN to detect drowsiness; focuses on eyes
Harada et al. [12]	Calculates pupil diameter from eye-tracking data; RNN predicts distraction levels

III. DRIVER DROWSINESS DETECTION SYSTEM DESIGN

Fig. 1 shows the system diagram and the system functions as follows. The camera captures a frame that marks the initial event. This image, stored in memory, is processed through a series of pre-processing steps to ensure the image is in the appropriate size for the model and similar to what the model was trained on. These steps include but are not limited to thresholding and grayscale conversions.

Additionally, it is desired for the driver's head to be located and a rectangle surrounding said head to be extracted from the image. This eliminates any background and primarily focuses on the driver's head movements. Once the pre-processing steps are completed, the image is transferred to the first model, where a label is given to the image. This label is saved in the database. Fig. 2 shows an example of a database for driver states.

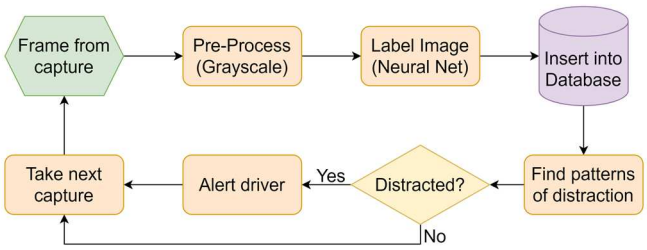


Fig. 1. System Flowchart for Driver Attention Monitoring

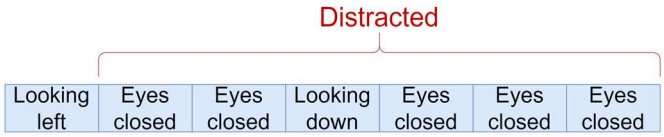


Fig. 2. Example of Database for Driver States

In Fig. 2, each block represents a fixed timestamp. The first block can be at one unit of time, the second block at one unit of time, the third block at one unit of time, and so on. This database is an array containing all previous and current states of the driver. A fixed amount is fed into the second model to output whether the driver is distracted or not. If the model finds irregularities, the system alerts the driver through mechanical stimulations—either a mechanical buzzer or sound using a speaker. Finally, a new camera frame is captured, and the program pre-processes the image again.

IV. HARDWARE DESIGN

The hardware consists of a 3-channel camera and a single-board computer. The model of the camera used is LT-IMX219-MIPI-FF-NANO-H136 V1.3 [13]. The resolution of this camera is 8.08 megapixels. The Intel RealSense Depth Camera D435i can also be considered as a method of using a depth map to aid the program for our system.

Several single-board computers and one proprietary system were examined as possible platforms. Table II shows a table comparing these single-board computers. They are all within a price range of around $100. The Nvidia Jetson Nano 4GB was selected as the best fit for this project. The other two single-board computers were the Raspberry Pi 4 and the Google Coral Development Board. All three were comparable in physical dimensions, and no concerns were raised in terms of the system's feasibility. From a power consumption perspective, the Jetson Nano would draw 10 watts at peak performance, the Raspberry Pi consumes around 6 watts, and the Google Coral Development Board consumes around 2 watts.

TABLE II. HARDWARE COMPARISON FOR DRIVER DROWSINESS DETECTION

	Jetson Nano 4GB	Raspberry Pi 4	Coral Dev Board
Power Consumption	10W	5W	2W
CPU	Quad-Core Arm Cortex-A57	Quad-Core Cortex-A72	Quad-Core Cortex-A53
GPU	128-Core Maxwell	VideoCore VI	GC7000 Lite
TPU	N/A	N/A	Google Edge TPU
RAM	4 GB	8 GB	1 GB
Computation Power	472 GFLOPs	9.69 GFLOPs	64 GFLOPs

The NVIDIA Jetson Nano 4GB requires 10W of power (5V @ 2A) to operate at its fullest. There are multiple ways to supply 10W of power from a vehicle without making modifications, thus it was determined that there were no power limitations to the computers to be examined. The system is designed as a module plugged into an external power supply; batteries were not in this project's scope and were not looked at as an option. Comparing the three computers, all possessed a CPU, yet only the Jetson Nano and Google Coral Development Board were equipped with a GPU. The purpose and benefit of having a GPU are to compute tedious calculations significantly faster than running the same processes on a CPU. Machine learning can be considered a 'net' of tedious computations. The Jetson Nano became the final choice for our system design as it outperformed the Google Coral Development Board as shown in Table II.

V. SOFTWARE DESIGN

A. Design Criteria

A few design criteria were in place to aid the design process to fulfill the functional expectations of the system. One of the most significant issues regarding monitoring systems is privacy. Our system shall have a closed-loop camera feed, where the taken images will remain in memory only on the Jetson Nano. Furthermore, we require a system to take all our training videos and pre-process them into the correct format. For this purpose, we need to design a streamlined system, ideally using a single script to handle this. This is because we want to compare data types and pre-processing methods to achieve the most accurate model.

B. AI Algorithm and Software Tools

Two libraries were examined as part of the project's scope: TensorFlow and PyTorch. TensorFlow is an open-source library for machine learning and artificial intelligence. With uses in all areas due to its implementation of tensors, its primary focus resides in the training and inference of deep neural networks. PyTorch, on the other hand, is commonly used for applications involving computer vision and natural language processing. TensorFlow and PyTorch are sought after due to their extensive documentation, uses, and available functions. However, TensorFlow was the chosen library for this system design due to the numerous models available online due to the age of the library. PyTorch was a great competitor and performed very well, however, there were technical issues with running our models on the GPU of the Jetson Nano when using PyTorch.

C. Feature Extraction Model

Our program contains two models; the former is responsible for feature extraction. With an input of a frame from the camera, its output matches one of six states: eyes open, eyes closed, looking up, looking down, looking left, or looking right. The frame goes under pre-processing operations as described. The frame is a 3-channel image, with each channel corresponding to a color channel. The frame is converted to grayscale, and a flattening operation is then used to create an array that can be inputted into the model. A custom dataset was constructed for this model.

Additionally, an eye cascade file was used as a part of OpenCV to aid in detecting the eyes of the driver. The model was trained on the custom dataset as it created a model that would best fit a specific driver. Utilizing approximately 3,000 and 500 frames respectively for the training and testing set, we were able to achieve an accuracy of 95% during training. All output of this model was saved to a database for use in the second model, Driver State Determination.

D. Driver State Determination Model

Every driver has a different driving style and pattern. The driver state determination model in the program was responsible and designed to learn these patterns of a specific driver. There are many unique case scenarios, such as the time one looks over their shoulder when changing lanes. This leads us to need a machine learning model to be implemented to account for these patterns. This model was designed to take a segment of the database, as shown in Fig. 2 previously, as the input. The output is a Boolean expression of whether the driver is distracted.

E. Time Constraints of Model

The speed at which the program executes needs to be near real-time as a part of the project requirements—the significance of such lies in what happens if an unexpected event occurs during operation. If the program performance were to lag for any reason or reset itself, the model could lack data that could indicate a distracted driver. If this were ever the case, the system would fail to alert the driver, and an accident could occur. Our program operated in a range of 45 to 60 frames per second. No lags or unexpected events occurred during our testing, but it does not guarantee that our system is protected against such.

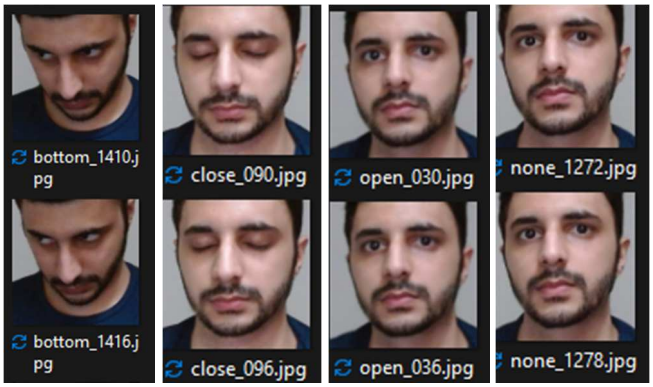
F. Security

A significant concern in many systems available on the market is privacy. The system developed in this paper functions on a closed-loop feed, meaning the camera feed is only given to the Jetson Nano for processing and then discarded. There is no Wi-Fi or Bluetooth connection, and no images or driving patterns are saved onboard the single-board computer, resting privacy concerns as there is no possibility for any data to be leaked as a part of the code.

VI. RESULTS

Regarding results, we are focused on two different libraries and on how each effectively assists us in developing and training the model. Furthermore, based on prior states, we have

identified a reliable technique for identifying a distracted driver. We assembled the dataset to produce the results and trained the network within days. To choose and finish the library for the model, one team member worked in TensorFlow while the other used PyTorch. However, PyTorch was not considered as there were some functionality problems. Data will be collected individually and compiled into one database that will be shared with everyone to ensure consistent results. The following figure, Fig. 3, shows seven directories being created after running the script for the first time. The following figures display different training frames stored in the training frame directory after extracting the frames from the video file using OpenCV. The different stored frames include eyes opened and closed, looking up, down, left, and right.



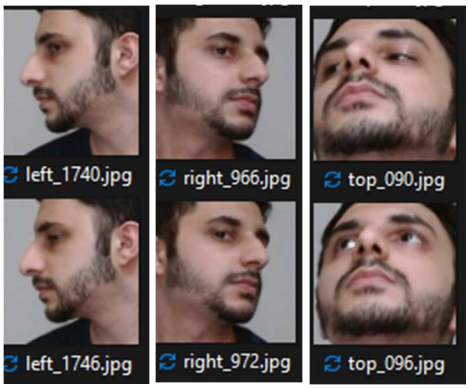


Fig. 3. Stored Frames for Training and Testing

Fig. 4 displays the terminal output as it currently displays the action either none, which means looking straight, or right, meaning looking right. It also displays the current time, and if eyes are opened or closed, if the driver is distracted, it will show the driver distracted, and a sound will also play in the preview.

```
eye: opened
stop sound ++++++++++++++++++++++++++++++++++++++++++++
Date and time is: 2022-07-19 20:33:28.376330
Timestamp is: 1658280808.37633
1/1 [==============================] - 0s 36ms/step
action :  none
eye: opened
stop sound ++++++++++++++++++++++++++++++++++++++++++++
Date and time is: 2022-07-19 20:33:28.535853
Timestamp is: 1658280808.535853
1/1 [==============================] - 0s 45ms/step
action :  right
```

Fig. 4. Terminal Output of Driver Attention Monitoring

Fig. 5 displays different actions none (looking straight), left, right, up, and down as well as if eyes are opened and closed.

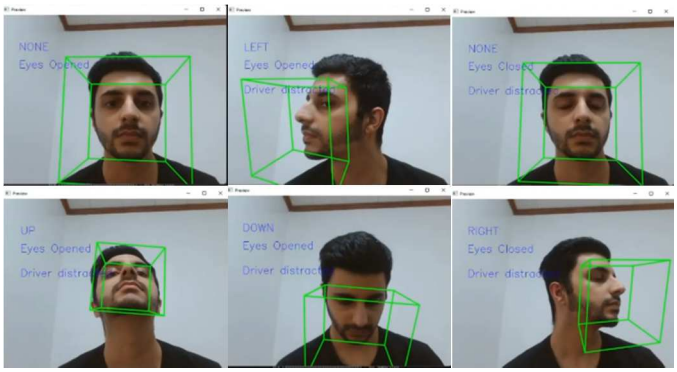


Fig. 5. Examples of Driver Facial Actions

VII. System Design Discussion

With the current state of the system and the classification of this topic being open-ended, there are many opportunities to build off this system and advance its features. In terms of what else can be added, a system that lets drivers add their requirements into the system according to their behavior when behind the wheel would be preferable. Also, other features including head position and how the driver uses the steering wheel can be added, as well as the speed of the driver as some young drivers and older drivers are the leading cause of accidents due to either being too hyper or not experienced enough. For older drivers, not having enough reaction time while driving is an issue worth considering.

Further discussion is also to be had on how we can develop a system that follows the same process. Furthermore, it enhances and utilizes the second model we had proposed but not implemented, the pattern network model, which will determine the requirements and behavior of that user and may even identify the user through facial recognition. As well as the depth camera D435i, which can help us further enhance and improve the dataset. When the camera moves, adding an IMU enables the application to improve its depth perception. This camera allows for simple monitoring and SLAM applications that enable better point-cloud alignment. Additionally, it enables the system to identify any driving motions, even in low light, thanks to better environmental awareness.

The Feature Extraction Model has many optimization opportunities available, such as adopting a more efficient and

calculated neural network structure. The goal is to discard the eye cascade file as it slows the program down due to its inefficient method of finding patterns. An alternative YOLO (You Only Look Once) algorithm could be applied, yet it would only complicate and buckle down the code compared to a singular or even multiple deep neural network that can run on the GPU.

VIII. Conclusion

As this is an open-ended area of work, there are many solutions to different problems, and each solution can build off each other to further advance driver monitoring systems. This paper discussed the importance of driver attention monitoring systems and the method that was carried out to create a system that functioned to a certain extent. Many improvements can be made, and this paper serves as a baseline for future work and for others to build off the ideas in this paper. The goal is for a refined and renewed version of this code to be completed and tested in a real-world situation.

References

[1] "Distracted driving," Centers for Disease Control and Prevention, 26-Apr-2022. [Online]. Available: https://www.cdc.gov/transportationsafety/distracted_driving/. [Accessed: 12-Jun-2022].

[2] K. Barry, "Tesla's camera-based driver monitoring fails to keep driver attention on the road, CR tests show," *Consumer Reports*, 22-Dec-2022. [Online]. Available: https://www.consumerreports.org/car-safety/tesla-driver-monitoring-fails-to-keep-driver-focus-on-road-a3964813328/

[3] J. S. Wijnands, J. Thompson, K. A. Nice, G. D. P. A. Aschwanden, and M. Stevenson, "Real-time monitoring of driver drowsiness on mobile platforms using 3D Neural Networks - neural computing and applications," *SpringerLink*, 13-Oct-2019. [Online]. Available: https://link.springer.com/article/10.1007/s00521-019-04506-0. [Accessed: 09-Jun-2022].

[4] J. 19 and P. Subramanian, "Nvidia Drive IX keeps drivers focused on the road ahead," *NVIDIA Blog*, 19-Jan-2021. [Online]. Available: https://blogs.nvidia.com/blog/2021/01/19/drive-ix-ai-software-drivers-safe/. [Accessed: 09-Jun-2022].

[5] K. Barry, "Tesla's camera-based driver monitoring fails to keep driver attention on the road, CR tests show," *Consumer Reports*, 22-Dec-2022. [Online]. Available: https://www.consumerreports.org/car-safety/tesla-driver-monitoring-fails-to-keep-driver-focus-on-road-a3964813328/

[6] P.-heng Hong and Y. Wang, "Real-time driver's focus of attention extraction and prediction using Deep Learning," *International Journal of Advanced Computer Science and Applications (IJACSA)*, 2021. [Online]. Available: https://thesai.org/Publications/ViewPaper?Volume=12&Issue=6&Code=IJACSA&SerialNo=1. [Accessed: 09-Jun-2022].

[7] Jiménez R, Avilés O, Amaya D (2014) Driver distraction detection using machine vision techniques. *Ing Compet* 16(2):55–63. https://doi.org/10.25100/iyc.v16i2.3683

[8] Ying Y, Jing S, Wei Z (2007) The monitoring method of driver's fatigue based on neural network. In: *Proceedings of the 2007 IEEE international conference on mechatronics and automation*, IEEE, Harbin, pp 3555–3559. https://doi.org/10.1109/ICMA.2007.4304136

[9] Ribarić S, Lovrenčić J, Pavešić N (2010) A neural-network-based system for monitoring driver fatigue. In: *Melecon 2010–2010 15th IEEE Mediterranean electrotechnical conference*, IEEE, Valletta, pp 1356–1361. https://doi.org/10.1109/MELCON.2010.5475993

[10] Ji Q, Zhu Z, Lan P (2004) Real-time nonintrusive monitoring and prediction of driver fatigue. *IEEE Trans Veh Technol* 53(4):1052–1068. https://doi.org/10.1109/TVT.2004.830974

[11] Jiangwei C, Lisheng J, Bingliang T, Shuming S, Rongben W (2004) A monitoring method of driver mouth behavior based on machine vision. In: *2004 IEEE intelligent vehicles symposium*, IEEE, Parma, pp 351–356. https://doi.org/10.1109/IVS.2004.1336408

[12] Rong-ben W, Ke-you G, Shu-ming S, Jiang-wei C (2003) A monitoring method of driver fatigue behavior based on machine vision. In: *Proceedings of IEEE IV2003 intelligent vehicles symposium*. IEEE, Columbus, OH, pp 110–113. https://doi.org/10.1109/IVS.2003.1212893

[13] Harada T, Iwasaki H, Mori K, Yoshizawa A, Mizoguchi F (2013) Evaluation model of cognitive distraction state based on eye-tracking data using neural networks. In: *Proceedings of the 12th IEEE international conference on cognitive informatics and cognitive computing*, IEEE, New York, NY, pp 428–434. https://doi.org/10.1109/ICCI-CC.2013.6622278

[13] Leopard Imaging Inc, "LI-IMX219-MIPI-FF-NANO SPECIFICATION", 26. Apr. 2019.