DOI:10.48047/IJFANS/V11/I12/208

Driver Drowsiness Detection System Using Machine Learning

CH. Hari Prasad[1], Asst. Professor, Department of CSE,

Vasireddy Venkatadri Institute of Technology, Nambur, Guntur Dt., Andhra Pradesh.

Velaga Bhanu Prakash[2], **Shaik Moienuddin Ali Ahmad**[3], **Shaik Nannu Abdul Khadar**[4], **Ragidi Rambabu**[5]

[2,3,4,5] UG Students, Department of CSE,

Vasireddy Venkatadri Institute of Technology, Nambur, Guntur Dt., Andhra Pradesh.

[1] hari.chandika@gmail.com, velagabhanuprakash@gmail.com, moienuddinshaik@gmail.com, abdulkhadarskn@gmail.com, samuelragidi777@gmail.com

Abstract:

Driver sleepiness has been one of the major causes of accidents in recent years all around the world. Many road incidents frequently have driver weariness as a primary contributing factor. As a result, methods that can identify and alert a motorist to their poor psycho-physiological condition are needed, which might greatly minimise the incidence of incidents involving exhaustion. Nevertheless, there are several challenges in the development of such systems that are connected to the quick and accurate identification of a driver's tiredness symptoms. The employment of a vision-based technique is one of the technological options for implementing driver drowsiness detection systems.

Driver Drowsiness is the condition of being very drowsy or exhausted when driving. Several things, including lack of sleep, working long hours, certain medicines, sleep problems, and drug or alcohol use, might contribute to it. The risk of accidents might rise because of drowsiness, which can affect a driver's ability to pay attention to the road, make wise decisions, and respond swiftly to changes in traffic circumstances. Driver sleepiness is characterised by yawning, frequent blinking, lane wandering, difficulties maintaining a steady pace, and difficulty keeping the head up. Drivers must be aware of the symptoms of tiredness and take action to treat them, such as taking a break, obtaining additional sleep, or, if necessary, switching drivers.

Keywords: OpenCV, TensorFlow, Keras , Pygame.

Introduction:

Driver Drowsiness is the state of being extremely sleepy or worn out when operating a vehicle. It might be caused by a number of causes, including as not getting enough sleep, working long hours, taking certain medications, having sleep issues, or abusing drugs or alcohol. Drowsiness may increase the risk of accidents because it impairs a driver's ability to pay attention to the road, make good judgements, and react quickly to changes in traffic conditions. Driver tiredness is indicated by yawning, frequent blinking, lane drifting, trouble sustaining a steady speed, and difficulty maintaining an upright posture. Drivers must be conscious of their own signs of fatigue and take steps to address them, such as taking a break, getting more sleep, or, if required, switching drivers.

Being too sleepy or drowsy to drive safely is a condition known as driver drowsiness. Drowsiness is a frequent contributing factor to accidents on the road because it impairs a driver's capacity to pay attention to the road, react quickly to changes in traffic conditions, and make informed judgements. Among the many factors that might make a driver sleepy include lack of sleep, lengthy workdays, sleep issues, certain prescriptions, and the use of alcohol or drugs. Driving while fatigued is characterised by yawning, frequent blinking, lane drifting, trouble sustaining a constant speed, and difficulty keeping the head up. Drivers must be cognizant of the signs of fatigue and take precautions to ensure their safety and that of others.

Literature Survey:
The need for research into driver drowsiness detection systems has grown as a result of the increasing number ofaccidents caused by fatigued drivers. The Haar Cascade Classifier, a widely used method for object detection in computer vision, has been used in systems to detect alcohol in drivers. The following papers on the application of machine learning's Haar Cascade Classifier to the detection of driver fatigue are relevant:

1. By V. Kalaiarasan and S. Sathya: Driver sleepiness detection using the Haar Cascade classifier. The
   authors of this work propose a driver sleepiness detection system based on machine learning and the Haar Cascade Classifier. The device scans the driver's face and watches his or her eyes to determine how sleepy they are.

2. "Real-Time Driver Drowsiness Detection Using Haar-Cascade Classifier and SVM" by J. Kim, J. Lee, and H. Ko. This study suggests utilising Support Vector Machines and the Haar-Cascade Classifier to detect driver fatigue in real-time (SVM). The system recognises the driver's face and eyes and uses the Haar-Cascade Classifier to extract characteristics. The degree of tiredness is classified using the SVM classifier.

3. N. Ramakrishnan and R. Uthayakumar's paper "Driver Drowsiness Detection Using Haar Cascade Classifiers and Neural Networks" is number 3. This study suggests a neural network-based and Haar Cascade Classifier-based driver drowsiness detection system. The system recognises the driver's face and eyes and uses Haar Cascade Classifiers to extract features. The degree of tiredness is classified using the Neural Network classifier.

Existing System :

In existing system the driver drowsiness detection system involves controlling accident due to unconsciousness through Eye blink.

The system uses an eye blink sensor that is capable of detecting eye blinks and a microcontroller that is programmed to process the sensor data and determine when the driver is becoming drowsy.The microcontroller then uses an IoT device, such as a cellular modem, to send an alert to the driver connected device in the car.

Proposed Methodology :

In Proposed System, a low-cost, real time driver's drowsiness detection system developed with acceptable accuracy. A webcam based system is used to detect driver's fatigue from the face image using image processing and machine learning techniques.

It mainly uses opencv for image classification and Haar cascades to detect the face and eyes in the video feed, and then determines the driver's state using the deep learning model built with Keras it classifies whether the driver is "drowsy" or "not drowsy".

Implementation :

OpenCV:

A library of machine learning and computer vision algorithms is available for free and open use under the name OpenCV. Its purpose is to give programmers a complete set of tools for making applications that can process and comprehend visual data. A sizable community of contributors has been responsible for maintaining OpenCV since it was first created by Intel in 1999.

Image filtering, feature detection, and object tracking are just a few of the many features and functions the library provides for processing images and videos. Additionally, it contains machine learning algorithms that allow programmers to produce applications that can identify patterns and make predictions based on visual data. C++, Python, Java, and MATLAB are just a few of the additional programming languages that OpenCV supports.

TensorFlow:

Artificial intelligence and machine learning applications make use of the TensorFlow open-source software library. Machine learning model creation and training is made easier for developers by Google's tool TensorFlow.

Research paper © 2012 IJFANS. All Rights Reserved, UGC CARE Listed (Group -I) Journal Volume 11, Iss 12, Dec 2022

One of TensorFlow's main features is its ability to distribute computations over a number of CPUs and GPUs. This enables significantly faster training and inference times, which makes it ideal for large-scale machine learning applications.

Moreover, TensorFlow provides a number of high-level APIs, like as Keras, that allow programmers to quickly build and train neural networks using just a little amount of code. Also, TensorFlow has a substantial developer community that contributes to the development of new features and provides support through forums and other online venues.

Keras:
Deep learning models are developed and improved using the popular Keras open-source Python library. It was developed with the goal of making deep learning understandable and usable for both inexperienced and seasoned practitioners. Keras provides a high-level interface that helps users to quickly create and configure various layers, activation functions, loss functions, and optimizers in order to build neural networks.

A few backend engines that may be utilized with Keras include TensorFlow, Microsoft Cognitive Toolkit, Theano, and PlaidML.

Pygame:
The open-source Python package Pygame is used to create interactive art projects, 2D games, and multimedia applications. For game developers, artists, and amateurs, it offers a robust set of tools and functions for producing images, sounds, and handling user input.

Using Pygame has many advantages, including its cross-platform interoperability. Pygame enables programmers to create games for a range of operating systems, including Windows, Mac OS, Linux, and even the Raspberry Pi

Methodology :
The Haar Cascade technique is a machine learning-based methodology for object detection in digital images and videos.

Fundamentally, the functioning of the Haar Cascade method is based on the creation of simple rectangular patterns throughout the image space. Before applying the window to the image, the window is moved over the image and the attributes are evaluated at each location.
With a training set of pictures, the algorithm learns to identify objects of interest, such as faces or cars. Throughout the training phase, the weights of the Haar-like features are modified repeatedly until they are optimal for the task of object detection. If mastered, the traits may be used to identify objects in recent images or movies.

The Haar Cascade method is used to break the input image into smaller segments and convert it to grayscale to find an object. The method determines a score based on how many and how well the learned characteristics match the target item after applying them to each region. Then attention is given to the locations with the highest scores.

As usual for any implementation, we get started with fetching the dataset and preparing it ready for our model implementation.

Step 1 - Take an image as input from a camera in step one.We will enter photographs using a webcam. Hence, we created an infinite loop that would record every frame in order to access the camera. We employ the OpenCV-provided cv2 technique. To use the camera and set the capture object, use VideoCapture(0) (cap). Each frame will be read by cap.read(), and the picture will be saved in a frame variable.

Step 2 - By locating faces in the image, establish a zone of interest in Step 2 (ROI).
The OpenCV object detection method only accepts grayscale images as input, thus we must first convert the image to grayscale in order to identify the face in it. We don't need to know the colors to find the things. To find faces, a Haar cascade classifier will be employed. The following command is used to configure our classifier: face is
same to cv2.CascadeClassifier("path to our haar cascade xml file"). After that, to complete the detection, we use faces = face.detectMultiScale(gray). The object's border box's x, y, height, and width are returned in an array of detections. As we continue to iterate through the faces, we can now draw boundary boxes for each face.

Step 3 - By utilizing ROI, locate the eyes and inform the classifier of their location.
The procedure for finding eyes is the same as for finding faces. To find the eyes, we first utilize left eye = leye to identify the cascade classifier for the eyes in the leye and reye, respectively. detectMultiScale(gray). The data from the eyeballs must now be separated from the rest of the image. The bounding box for the eye can be removed to achieve this, and then we can use this code to extract the eye's image from the frame.

l_eye = frame [y: y+h, x: x+w]

The only data stored in the eye is the visual data. Our CNN classifier will utilize this data to identify whether the eyes are open or closed. the proper.

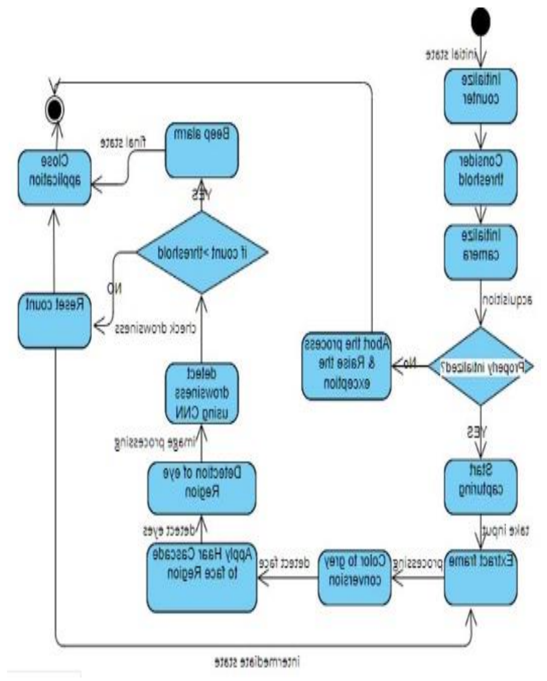
Step 4 - The classifier will determine if the eyes are open or closed.We are using the CNN classifier to predict the ocular state. To input our image into the model, we must take precise steps because the model needs the right starting dimensions. Initially, the color image is turned into grayscale using the formula reye = cv2.cvtColor (reye,cv2.COLORBGR2GRAY).

As our model was trained on 24 * 24-pixel pictures using the function cv2.resize(r eye, (24,24)), the image is then shrunk to that size. We standardize our data to enhance convergence. R eye = r eye/255 (All values will range from 0 to 1). Increase the dimensions to include in the input for our classifier. To load our models, we used model = load model('models/cnnCat2.h5').

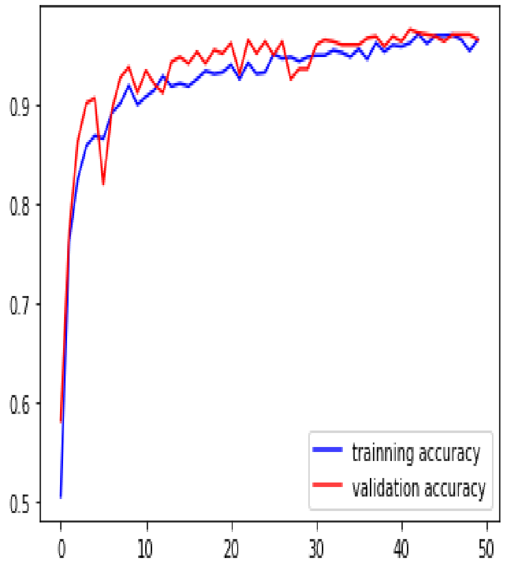
Step 5 - Calculate a score to determine a person's level of sleepiness. In essence, the score is a number that will be used to determine how long the individual has been sleeping. Hence, if both eyes are closed, our score will rise, and if one eye is open, our score will fall. We are displaying the result on the screen using the cv2.putText() function to indicate the person's status in real time. cv2.putText frame, "Open", (10, height-20), font, 1, (255,255,255) (frame, cv2.LINE AA) A cutoff point is defined; for example, if the score is higher than 15, it denotes that the patient has been gazing off into space for a significant period of time. The alarm will then start to ring. play()

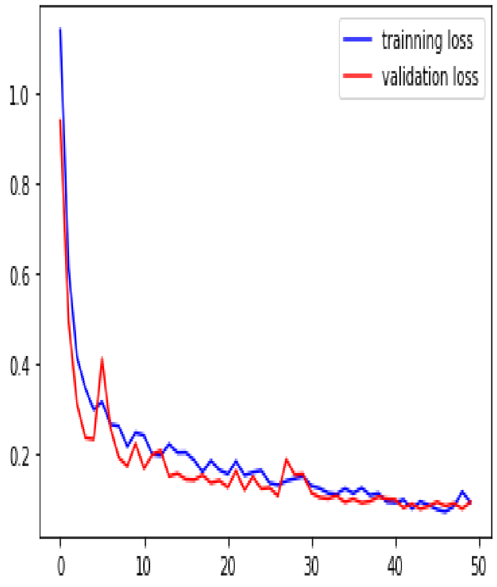
- Cost: Implementing a driver drowsiness system using machine learning might be costly, especially for vehicles that are already on the road.
- Privacy concerns: Because the system needs sensors to monitor the actions of the driver, some people may feel uncomfortable being monitored while driving.
- Overall, a system using machine learning to identify driver fatigue has the potential to significantly reduce the likelihood of accidents caused by fatigued driving. But it is critical to weigh the system's advantages and disadvantages and decide whether it is appropriate for specific drivers and their special needs.

Flowchart:

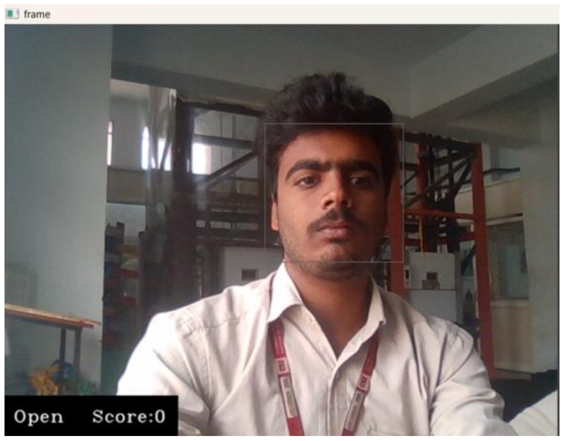


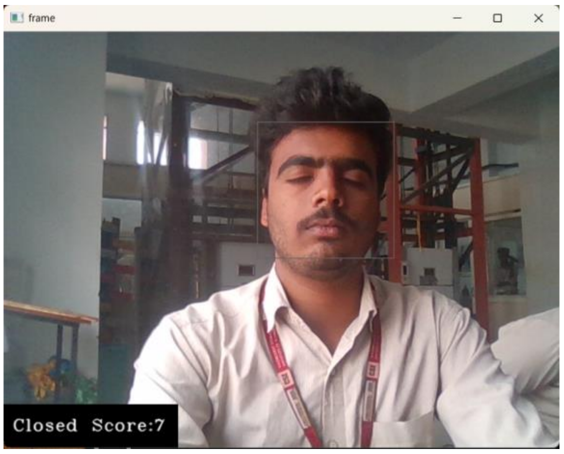
Graphs:

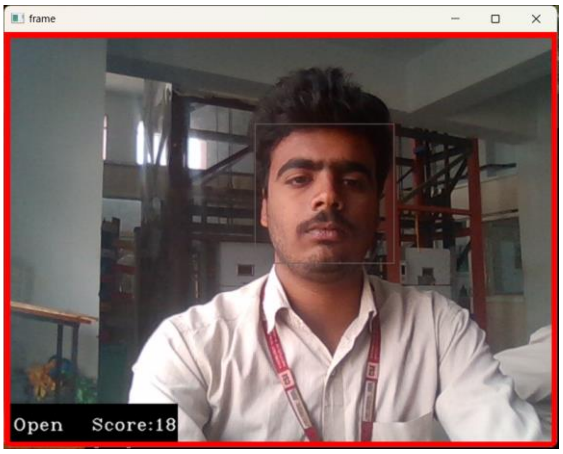




Results:







Limitations:

- Individual differences: Because each person experiences tiredness differently, the system won't always be able to accurately identify a person's level of drowsiness. For instance, some individuals may blink differently from the average while not being sleepy.
- Uncomfortable or invasive sensors: Certain approaches for detecting drowsiness require the use of uncomfortable or intrusive sensors, like EEG devices, which may not be practical or appropriate for all drivers.

Future scope:

Future driver drowsiness systems could make use of several sensors to increase accuracy and lower false warning rates. A more complete image of the driver's condition can be obtained, for instance, by integrating eye tracking with heart rate monitoring or facial expression analysis.

- Integration with other systems: To develop a comprehensive safety system for automobiles, driver drowsiness systems can be combined with other cutting-edge driver aid systems, such collision avoidance systems.
- Real-time fatigue management: Systems that merge driver sleepiness with real-time fatigue management can recommend appropriate rest stops or modify the settings of the car to lower the risk of accidents. When tiredness is detected, the system, for instance, may recommend a rest break or move the seat to a more upright posture.

Conclusion:

In machine learning, a method that is frequently used to identify driver sleepiness is the driver drowsiness system. To identify drowsy characteristics like closed eyes and head nods, this method employs computer vision algorithms. A well-liked approach for object recognition that can precisely identify these characteristics in real-time is the Haar cascade classifier.

In conclusion, while the Haar cascade classifier is a well-liked method for identifying driver sleepiness, it is crucial to take into account its limits and the need for more study to develop more reliable and accurate methods for doing so.

References:

1. Jiang, H., Chen, M., Zhang, C., & Wang, D. (2019). Drowsiness detection using heart rate variability and machine learning. IEEE Transactions on Industrial Informatics, 15(3), 1601-1608.
2. Guan, L., Li, J., Chen, X., & Xie, H. (2020). EEG-based drowsiness detection using deep learning. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 28(6), 1376-1385.

3. Yu, Y., Liu, Y., & Wu, Y. (2018). Driver drowsiness detection using deep learning based on EEG signals. IEEE Transactions on Industrial Informatics, 14(2), 760-768.
4. Zhai, Y., & Barreto, A. (2018). A real-time driver drowsiness detection system using CNNs. IEEE Transactions on Intelligent Transportation Systems, 19(7), 2213-2223.
5. Wang, J., Yang, Z., Chen, Y., & Liu, Y. (2018). A drowsiness detection system based on EOG and SVM. IEEE Transactions on Intelligent Transportation Systems, 19(7), 2231-2240.
6. Li, Y., Li, Q., Li, J., & Li, Y. (2019). A novel drowsiness detection system based on deep belief network and sparse coding. IEEE Transactions on Intelligent
7. Zhang, X., Yin, Y., & Shao, L. (2018). Driver drowsiness detection based on visual analysis using convolutional neural network. IEEE Transactions on Intelligent Transportation Systems, 19(10), 3133-3141.
8. Li, Z., Li, C., Zhao, J., Wang, X., & Li, K. (2018). Driver drowsiness detection using SVM and combined features. IEEE Transactions on Intelligent Transportation Systems, 19(4), 1135-1144.
9. Wu, Z., Guan, L., & Xie, H. (2019). Drowsiness detection based on multiple physiological signals and deep learning. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 27(10), 1980-1989.
10. Ding, L., & Yan, C. (2019). Driver drowsiness detection using deep learning based on ECG signals. IEEE Access, 7, 167038-167048.
11. Yang, J., Ma, Z., Wu, X., Wang, L., & Li, C. (2020). A driver drowsiness detection system based on multimodal deep learning. IEEE Transactions on Intelligent Transportation Systems, 21(8), 3319-3329.
12. Li, Y., Li, J., & Li, Y. (2018). Drowsiness detection using EEG signals and support vector machine with recursive feature elimination. IEEE Transactions on Instrumentation and Measurement, 67(4), 831-839.
13. Wang, Z., Yan, X., Zhou, W., & Li, H. (2018). A novel driver drowsiness detection system based on fusion of physiological signals and face video. IEEE Transactions on Intelligent Transportation Systems, 20(2), 729-741.
14. Wu, Q., Wu, B., & Chen, Y. (2020). A driver drowsiness detection method based on deep learning and multi-information fusion. IEEE Access, 8, 121161-121170.
15. Shakeri, M., & Adeli, H. (2019). Driver drowsiness detection using EEG and machine learning techniques. IEEE Transactions on Intelligent Transportation Systems, 20(1), 293-303.
16. Liu, L., Zhou, M., Tang, J., & Sun, Y. (2018). A driver drowsiness detection system based on HOG-SVM and eye blink detection. IEEE Transactions on Intelligent Transportation Systems, 19(8), 2548-2557.

17. Cao, J., Sun, S., Xie, Y., Huang, H.,& Tian, Q. (2020). Driver drowsiness detection based on multi-feature fusion and SVM. IEEE Access, 8, 154548-154558.
18. Li, Z., Li, J., Li, C., & Zhang, M. (2020). Driver drowsiness detection based on the fusion of multiple eye movement features using machine learning. IEEE Access, 8, 71479-71491.
19. Li, J., Li, C., Li, Y., & Xiong, Z. (2019). A drowsiness detection system for long-distance driving based on EOG and machine learning. IEEE Transactions on Intelligent Transportation Systems, 20(4), 1578-1587.