https://doi.org/10.46335/IJIES.2023.8.10.2 e-ISSN: 2456-3463
Vol. 8, No. 10, 2023, PP. 6- 19

International Journal of Innovations in Engineering and Science, www.ijies.net

Driver Drowsiness Detection System using Arduino and Deep Learning

Omkar Jagtap[1], Somesh Chaurasia[2], Pranjali Choudhary[3], Gopal Buwade[4], Mahesh Dhote[5], Dr. U.B. Aher[6]

[1-5]Student, Department of Computer Engineering, Government Polytechnic, Nagpur.
[6] Lecturer, Department of Computer Engineering, Government Polytechnic, Nagpur.

omkarjagtap1011@gmail.com

***Received on**: 25 October, 2023* ***Revised on**: 24 November 2023* ***Published on**: 28 November, 2023*

***Abstract--** The proposed paper aims to develop a nonintrusive real-time fatigue detection system, with a specific emphasis on long-distance drivers. This system employs camera technology to capture images continuously and assesses the driver's eyes through a specified algorithm. The algorithm focuses on identifying frames with closed eyes as indicative of fatigue. Upon detecting a closed eyes frame, a counter is initiated, incrementing with each subsequent identification. Once the counter surpasses a predefined threshold, an alert is triggered. Conversely, if open eyes are detected, the counter resets to zero. The system incorporates machine learning algorithms, ensuring robust performance across various driving conditions. The algorithm leverages a diverse dataset to refine its ability to discern subtle signs of fatigue, making it adept at detecting nuanced patterns indicative of drowsiness. This dynamic learning capability positions the system as an intelligent and evolving solution, capable of addressing the evolving challenges associated with driver fatigue. As a result, the proposed project not only aligns with current safety standards but also anticipates future advancements, contributing to a safer and more secure driving environment.*

***Keywords:** Fatigue Detection System, Open CV, Convolutional Neural Network (CNN), Driver Drowsiness, Face Detection.*

I-INTRODUCTION

1.1 Problem Definition

Driver drowsiness poses a critical threat to road safety, necessitating the development of an effective alert system specifically designed for four-wheeled vehicles. The current landscape relies heavily on wearable devices that utilize vibrations to alert drivers, yet these solutions are marred by inherent limitations, including user irritation and the inconvenience of having to wear them. This research addresses these shortcomings and endeavours to create a non-intrusive drowsiness detection system tailored for all drivers of four-wheelers.

The proposed system seeks to leverage deep learning models integrated with cameras, LEDs, buzzers, and LCDs to detect signs of drowsiness, such as closed eyes or the onset of sleep. Unlike existing wearable, this system aims to provide real-time alerts without requiring the driver to wear any additional devices. This approach not only enhances user comfort but also expands the reach of the solution to a wider demographic of drivers.

However, implementing an efficient drowsiness detection system using these components poses several challenges, most notably the need to minimize false alarms. False alerts

https://doi.org/10.46335/IJIES.2023.8.10.2 e-ISSN: 2456-3463
Vol. 8, No. 10, 2023, PP. 55- 58

can erode user trust in the system and may lead to disregard or disuse. This research focuses on refining the deep learning model to achieve a high level of accuracy in distinguishing between genuine drowsiness indicators and normal variations in driver behavior. By addressing this challenge, the system aims to become a reliable and non-intrusive tool in enhancing road safety.

Additionally, the project recognizes the importance of real-world applicability. Testing scenarios will be diverse, encompassing various driving conditions and environmental factors, to ensure the robustness and adaptability of the system. This research aims to contribute to the advancement of drowsiness detection technology, not only by introducing a novel and non-intrusive alert system but also by addressing the critical issue of minimizing false alarms, thereby promoting a safer driving experience for all.

1.2 Motivation

Road safety has emerged as a critical global concern, with traffic accidents claiming the lives of millions annually. Among the various factors contributing to road accidents, driver drowsiness stands out as a significant and often underestimated threat. Drowsiness-related incidents account for an estimated 20% of all road accidents worldwide, with this figure rising to as high as 50% on certain roads. The consequences of drowsy driving can be devastating, leading to fatalities, injuries, and substantial property damage.

The Samruddhi Mahamarg, a newly constructed expressway in Maharashtra, India, has witnessed a concerning rise in accidents attributed to driver drowsiness. This expressway, with its long stretches of highway and high speeds, presents a particularly dangerous environment for drowsy drivers. The monotonous nature of long-distance driving and the reduced alertness associated with fatigue can significantly impair a driver's ability to react promptly and make sound decisions, increasing the risk of accidents.

The alarming frequency of drowsiness-induced accidents on the Samruddhi Mahamarg highlights the urgent need for a reliable and effective system to detect and alert drivers of their drowsy state. In 2022, the Samruddhi Mahamarg recorded a staggering 358 accidents, with driver fatigue identified as a major contributing factor in 27% of these incidents. The impact of these accidents is far-reaching, not only causing loss of life and injuries but also leading to significant economic losses due to property damage and disruptions in transportation.

1.3 Objective

The overarching goal of this project is to develop a sophisticated and unobtrusive drowsiness detection system that employs a camera and a buzzer to identify and alert drivers of their drowsy state. The system should exhibit exceptional accuracy in differentiating between open and closed eyes, minimizing false alarms to avoid unwarranted distractions for the driver. To achieve this objective, the MobileNetV1 model is modified to create a customized model specifically tailored for drowsiness detection. This customized model utilizes a deep learning approach to analyze the driver's facial features, particularly eye states, to accurately identify signs of drowsiness.
Specific objectives encompass:

1. **Real-time Eye Detection:** The system must effectively detect and track the driver's eyes in real time, even under varying illumination conditions and with diverse facial expressions.
2. **Precise Eye State Classification:** The customized MobileNetV1 model should accurately classify the driver's eye state as open, closed, or partially closed, providing a reliable assessment of their drowsiness level.
3. **Timely and Effective Alerts:** The system should provide timely and effective alerts to the driver when drowsiness is detected, utilizing visual and auditory cues.
4. **Minimization of False Alarms:** The system should be meticulously designed to minimize false alarms, ensuring that alerts are triggered only when genuine drowsiness is detected, avoiding unnecessary distractions for the driver.

1.4 Scope of the Study

The study conducted is sharply focused on developing and implementing a non-intrusive Driver Drowsiness Detection System specifically designed for the unique demands of the Samruddhi Mahamarg. What sets the proposed project apart is its commitment to simplicity and seamless integration into the vehicle environment. Unlike other products in the market that necessitate drivers to wear additional devices, the system stands out by utilizing commonplace components such as cameras and buzzers. This approach not only prioritizes accessibility but also eliminates the need for drivers to wear any extra components, enhancing user convenience. The scope extends to real-time drowsiness detection based on facial cues, with careful consideration given to potential challenges, including variations in lighting conditions and individual facial features.

https://doi.org/10.46335/IJIES.2023.8.10.2 e-ISSN: 2456-3463
Vol. 8, No. 10, 2023, PP. 55- 58

Recognizing these limitations, proposed study is dedicated to continuous refinement and updates to ensure the system's accuracy and reliability. Through this endeavour, the aim is to contribute a practical, user-friendly solution to address drowsy driving and improve road safety on the Samruddhi Mahamarg.

1.5 Project Significance

The successful implementation of this drowsiness detection system holds immense potential to enhance road safety on the Samruddhi Mahamarg and beyond. By effectively detecting and alerting drowsy drivers, the system can significantly reduce the incidence of fatigue-related accidents, saving lives and preventing injuries. Additionally, the unobtrusive nature of the camera-based approach is expected to increase the system's acceptability among drivers.

The development of this drowsiness detection system aligns with the growing emphasis on advanced driver-assistance systems (ADAS) that aim to improve road safety by reducing human error. By addressing the critical issue of driver drowsiness, this project contributes to the broader goal of promoting safer and more responsible driving practices.

II- EXISTING SYSTEM

Several existing fatigue detection systems rely on hardware-based sensors and modules to monitor physiological and behavioural indicators of fatigue. While these approaches offer direct measurements of fatigue-related signals, they often face limitations in terms of practicality, cost, and reliability.

Infrared Sensor-Based Systems:

One common approach utilizes infrared (IR) sensors placed on the eye to detect fatigue-related changes in pupil dilation. However, this method requires direct contact with the user's eye, which can be uncomfortable and impractical for everyday use. Additionally, loose sensor placement can lead to inaccurate measurements and false detections.

EEG-Based Systems

Electroencephalography (EEG) systems measure brain activity to assess fatigue levels. While EEG provides rich information about brain state, it requires complex and expensive equipment, making it impractical for widespread use. Moreover, EEG signals can be influenced by various factors, potentially affecting the accuracy of fatigue detection.

EEG-Based Systems

Electroencephalography (EEG) systems measure brain activity to assess fatigue levels. While EEG provides rich information about brain state, it requires complex and expensive equipment, making it impractical for widespread use. Moreover, EEG signals can be influenced by various factors, potentially affecting the accuracy of fatigue detection.

Wearable Device-Based Systems

Wearable devices, such as smart watches or fitness trackers, have gained popularity for fatigue monitoring.

However, these devices often rely on indirect measures of fatigue, such as heart rate variability or activity levels, which may not always accurately reflect fatigue state.

Challenges with Hardware-Based Fatigue Detection

- The use of hardware-based sensors and modules for fatigue detection presents several challenges:
- Cost: The requirement for specialized hardware can
- Significantly increase the overall cost of the system, limiting its accessibility.
- Usability: Hardware sensors often require direct contact with the user's body, which can be uncomfortable and impractical for everyday use.
- Reliability: Hardware connections can become loose or malfunction, leading to false detections and reduced system reliability.
- Setup Complexity: Assembling and configuring the hardware components can be a time-consuming and tedious process, requiring technical expertise.

III-PROPOSED SYSTEM

The proposed system is a non-intrusive drowsiness detection system that utilizes a camera, a buzzer, and a processing unit to identify and alert drivers of their drowsy state. The system effectively detects and tracks the driver's eyes in real time, even under varying illumination conditions and with diverse facial expressions. The customized MobileNetV1 model accurately classifies the driver's eye state as open, closed, or partially closed,

https://doi.org/10.46335/IJIES.2023.8.10.2 e-ISSN: 2456-3463
Vol. 8, No. 10, 2023, PP. 55- 58

providing a reliable assessment of their drowsiness level. When drowsiness is detected, the system promptly alerts the driver through visual and auditory cues, maximizing the chances of preventing a drowsy driving incident.

Advantages of the Proposed System are:

- Non-intrusive: The system does not require drivers to wear any additional devices, enhancing user comfort and acceptance.
- Real-time detection: The system operates in real time, providing immediate feedback to drivers about their drowsiness state.
- High accuracy: The customized MobileNetV1 model demonstrates high accuracy in classifying eye states, minimizing false alarms and maximizing the system's reliability.
- Dual-alert mechanism: The combination of visual and auditory alerts ensures that drivers receive clear and timely warnings about drowsiness.
- Seamless integration: The system is designed for easy integration into the vehicle environment, with discreet camera placement and compatibility with existing audio systems.

Overall, the proposed drowsiness detection system offers a promising solution for enhancing road safety by addressing the critical issue of driver fatigue. Its non-intrusive nature, real-time detection, high accuracy, dual-alert mechanism, and seamless integration make it a valuable tool for promoting safer driving practices and reducing the risk of drowsy driving accidents.

IV-LITERATURE SURVEY

A variety of methodologies proposed by researchers for the detection of drowsiness and blinking in recent years are:

4.1 Driver Drowsiness Detection [1]

The system consists of two main components: image processing and drowsiness calculation. The image processing component uses computer vision techniques to detect and track the driver's eyes in real-time. The drowsiness calculation component uses eye closure duration, yawning frequency, and head nodding to calculate the driver's drowsiness level. If the driver's drowsiness level exceeds a certain threshold, an alarm will be generated to alert the driver and prevent any potential accidents.

The image processing component is implemented using Python, Open CV, and dlib libraries. The system captures images from a camera mounted in front of the driver and uses a Haar Cascade classifier to detect the driver's face. The dlib library is then used to detect and track the driver's eyes in real-time by detecting the location of the pupils in the eye region. The drowsiness calculation component uses several metrics to calculate the driver's drowsiness level. Eye closure duration is calculated by measuring the time between consecutive eye blinks. Yawning frequency is detected by monitoring the size of the mouth opening. Head nodding is detected by tracking the position of the head in the camera frame. These metrics are combined using a weighted algorithm to calculate the driver's drowsiness level. If the driver's drowsiness level exceeds a certain threshold, an alarm is generated to alert the driver. The alarm can be in the form of an audible alarm or a visual signal, such as a flashing light. This system also logs the drowsiness level and the time of detection for future analysis and evaluation.

This system provides a reliable and efficient solution to detect and prevent driver drowsiness. The system can be easily integrated with any vehicle and can potentially save lives by preventing accidents caused by driver drowsiness.

4.2 Real-Time Fatigue Detection System using OpenCV and Deep Learning [2]

Nowadays, the Driver's safety in the car is one of the most desired systems for avoiding accidents. The goal for the paper is to ensure the security system. A system that can check the Driver's condition for fatigue and alert the Driver before it is too late is desired. In the proposed system, a driver assistance system using a camera is used that will focus on the open or closed state of the Driver's eyes by monitoring the state of the eyes, Calculating the EAR ratio continuously for each frame detection drowsiness is easy. In this technique, this will detect the Driver's fatigue state and alert the Driver using an alarm. Facial detection is achieved through OpenCV face detection. The eye ball is monitored for fatigue detection. The control unit controls every part in this system; if fatigue is detected, the system will give the alarm using the buzzer. Detection in real-time is the major challenge in the field of accident prevention systems.

4.3 Detecting Driver Drowsiness Based on Sensors: A Review [3]

Making an intoxicated driver operate a vehicle is unethical and unsafe. As a result, to conduct their studies, researchers have employed simulated environments. The key benefits of employing simulators include easy data gathering, safety, low cost, efficiency, and experimental control. The driving simulators can be broadly categorized as follows: Mid-level (Fixed-base) simulators with advanced imaging techniques, a large projection screen, a realistic car, and possibly a simple motion base; High-level (Motion-based) simulators with typically a view close to 360° and an extensive moving base; and Low-level simulators with a computer, a monitor,

https://doi.org/10.46335/IJIES.2023.8.10.2 e-ISSN: 2456-3463
Vol. 8, No. 10, 2023, PP. 55- 58

a realistic cockpit, a steering wheel, manual gear box, and pedals (clutch, brake, and accelerator).

Steering Wheel Movement (SWM) is a commonly used vehicle-based metric for determining the degree of driver fatigue. It is assessed using a steering angle sensor. A steering column-mounted angle sensor is used to measure the driver's steering behaviour. When driving while sleepy, there are less micro corrections made on the steering wheel than when driving normally. Sleep-deprived drivers did fewer steering wheel reversals than normal drivers, according to research by Fairleigh and Graham. The researchers only took into account little steering wheel motions (between 0.5° and 5°), which are required to modify the lateral position within the lane, in order to avoid the influence of lane changes. Therefore, using small SWMs, it is possible to assess the driver's level of drowsiness and, if necessary, issue a warning. To produce fluctuations in the lateral position and force the drivers to perform corrective SWMs, light side breezes were applied along a curving road in a simulated environment, pushing the automobile to the right side of the road. SWMs are used by automakers like Nissan and Renault, however they are only effective under certain circumstances. This is due to the fact that it can only operate dependably in specific conditions and is overly reliant on the geometric features of the road and, to a lesser degree, the kinetic properties of the vehicle.

4.4 Drowsiness Detection and Alert System: A Review [4]

The author of this project created goggles/spectacles equipped with IR sensors and a buzzer. The driver wears this full setup. A lot more components are included in the setup, including an Arduino UNO, a GSM SIM 800 module, two batteries, and two ON/OFF buttons that are connected to separate batteries. Here, the microcontroller is connected to the first battery, while the GSM module is attached to the second. The way the whole system operates is that, as soon as the driver puts on the goggles, infrared sensors detect whether or not the eyes are closed. If they are not, the system checks again, and so on, until the driver's eyes are discovered to be closed. When the eyes are discovered to be closed, the device performs a second check, and if the eyes are again found to be closed, the red LED and buzzer turn on and stay on for one minute before turning off. Following a minute, a condition is established whereby the driver is considered alert if the frequency is greater than fifty, and alert if the frequency is less than fifty. However, as soon as the driver is discovered to be sleepy, the owner receives a notification from the GSM module that says, "Driver is found drowsy." The same thing keeps happening again.

To put it succinctly, the driver's eyes begin to blink for longer than a second whenever they sense drowsiness. The IR sensor detects this condition, and the buzzer sounds to alert the driver and sends the owner an instant text message.

4.5 Facial features monitoring for real time drowsiness detection [5]

To train the classifier that will identify the object, the algorithm initially requires a large number of positive images—that is, images with faces—and negative images—that is, images without faces. Therefore, in addition to the Haar feature-based classifiers, the cascaded Adaboost classifier is used to identify the face region. After that, the compensated image is divided into a variety of rectangle areas that can be anywhere in the original image at any scale or position. Haar-like feature is effective for real-time face detection because of the variations in facial features. These can be computed using the difference in the sum of the pixel values within the rectangle. The Adaboost algorithm will accept all face samples and reject non-face samples of the images during this process.

4.6 Real time drowsiness detection using eye blink monitoring [6]

Using the Harris corner detection algorithm, which detects corners at the side and down curve of the eye lid, the image is first converted to greyscale in this method. Following the point-tracing process, a straight line will be drawn between the top and bottom points. The midpoint will then be connected to the lower point by the line's calculation. Now, it will follow the same steps for every image, calculating the 'd' distance between the midpoint and the lower point to determine the eye state. Ultimately, the computed distance 'd' is used to determine the eye state. The eye state is labeled as "closed" if the distance is zero or nearly zero; otherwise, it is labeled as "open." In order to determine whether the person is feeling sleepy or not, they have also called for intervals of time. This is accomplished by the average person blinking for 100–400 milliseconds, or 0.1–0.4 of a second. When the distance is zero or nearly zero, the condition of the eyes is categorized as "closed," otherwise, it is categorized as "open." In order to determine whether the person is feeling sleepy or not, they have also called for intervals of time. This is accomplished by the average person blinking for 100–400 milliseconds, or 0.1–0.4 of a second.

V-METHODOLOGY

5.1. Knowing Dataset

For the driver drowsiness detection system, the dataset is comprised of two fundamental categories of images: open

https://doi.org/10.46335/IJIES.2023.8.10.2 e-ISSN: 2456-3463
Vol. 8, No. 10, 2023, PP. 55- 58

eyes and closed eyes. This dataset serves as a pivotal component in training and validating the system's ability to recognize and respond to drowsiness in drivers. The "open eyes" images depict drivers with fully open, alert eyes, while the "closed eyes" images capture moments when drivers' eyes are partially or completely closed, signifying drowsiness or fatigue.

The use of open and closed eyes images enables the system to discern critical visual cues associated with drowsiness, such as eyelid drooping or prolonged eye closure, which are indicative of a driver's reduced attentiveness. These images are crucial for teaching the machine learning model to identify patterns and features that distinguish between an alert and drowsy state. As the system progresses, it can trigger warnings or interventions to prevent accidents by detecting signs of drowsiness in real-time, helping to ensure safer and more responsible driving behavior. This dataset forms the foundation of the drowsiness detection system, empowering it to enhance road safety and driver vigilance.

Table 4.1: Literature Survey

Title	Author	Faults	Conclusion
Driver Drowsiness Detection	Sundram Ohja, Ankita Agrawal	Sensor inaccuracies, false alarms	Improved system accuracy and reduced false positives
Real-Time Fatigue Detection System using OpenCV and Deep Learning	K Vijaychandra Reddy, Sanjana Gadalay	It is less efficient.	Suggests to develop more efficient model
Detecting Driver Drowsiness Based on Sensors: A Review	Arun Sahayadhas	Steering Wheel not detectable.	It is Impractical.
Drowsiness Detection and Alert System: A Review	Jyotsna Gabhane	Hardware malfunctions, user discomfort	Suggests more robust hardware design and cost optimization and user comfort
Facial features monitoring for real time drowsiness detection	Manu B.N	Limited dataset, slow response time	Recommends a larger dataset and faster processing algorithms
Real time drowsiness detection using eye blink monitoring	Amna Rahman	Environmental factors	Advocates for better adaptation to environmental conditions

The dataset consists of 50,000 images for each of the two essential categories: open eyes and closed eyes. These images are meticulously collected and manually cleaned to ensure high data quality. The dataset is thoughtfully organized, with images of open eyes stored in one folder and images of closed eyes stored in another.

The large quantity of images in each category provides a diverse and representative sample of real-world scenarios, enhancing the system's ability to generalize and accurately detect drowsiness in drivers across various conditions. Manual cleaning of the images involves removing any outliers, anomalies, or irrelevant data, guaranteeing the reliability and integrity of the dataset.

By maintaining separate folders for open and closed eyes images, the dataset is organized in a manner that facilitates efficient data management and access for the training and evaluation phases of the system. This careful curation and organization of the dataset, alongside its substantial size, plays a crucial role in the system's capacity to recognize the visual cues associated with drowsiness, thereby contributing to the overall success and safety of the driver drowsiness detection system.

Additional dataset is also used for the validation purpose. Images in this dataset are completely different than the training dataset. This dataset will be used for the validation purpose after the model is been trained after each epoch. The directory structure of this dataset is similar as that of training dataset. There are 5000 images of each category: open eyes and close eyes. [7]

5.2 Pre-processing Dataset

5.5.1. **5.5.1 Initializing Paths**

In order to preprocess the images in the dataset, the path needs to be specified from where the images will be accessed. So following variables are initialized specifying the path of the dataset.

dataset_directory: This variable is intended to store the path to the directory containing the training dataset.

validation_directory: Similar to dataset directory, this variable is meant to store the path to the directory containing the validation dataset.

5.5.2 preprocessing_image() function

This function is designed to process an input image before it is fed into the neural network. It first converts the image to grayscale using OpenCV's cv.cvtColor function with the

https://doi.org/10.46335/IJIES.2023.8.10.2 e-ISSN: 2456-3463
Vol. 8, No. 10, 2023, PP. 55- 58

COLOR_BGR2GRAY conversion. This is a common step in preprocessing to reduce the channel dimensionality of the image, making it easier to process. Subsequently, the image is converted back to RGB using cv.cvtColor with COLOR_GRAY2RGB. The reason for this conversion is to ensure that the input to the neural network has the same number of channels as expected. Finally, the image is resized to a target size using cv.resize. The resulting processed image is then returned. This preprocessing function is crucial for standardizing the input data and ensuring consistency in the format of images presented to the neural network.

5.5.3 Data Augmentation

An ImageDataGenerator object named datagen is instantiated. This object is part of Keras's image preprocessing utilities and is designed for real-time data augmentation during model training. The rescale parameter normalizes the pixel values of the images to the range [0, 1]. Various augmentation techniques are applied, including rotation (rotation_range), width and height shifting (width_shift_range and height_shift_range), shearing (shear_range), zooming (zoom_range), and horizontal flipping (horizontal_flip). The fill_mode parameter is set to 'nearest', indicating the strategy for filling in newly created pixels during data augmentation. Additionally, the preprocessing_function parameter is set to the previously defined preprocess_image function. This means that each image generated during training will undergo the specified augmentations as well as the custom preprocessing defined in preprocess_image.

5.5.4 Data Generators

In next part, two data generators (data_generator and validation_generator) are created using the flow_from_directory method of the ImageDataGenerator class. These generators retrieve batches of images from specified directories (dataset_directory and validation_directory). The target_size parameter ensures that all images are resized to a consistent size for input to the neural network. The batch_size parameter determines the number of images in each batch. The class_mode is set to 'binary', indicating that the classification task is binary (two classes). For the training data generator (data_generator), the shuffle parameter is set to True to randomize the order of data during training, enhancing the model's learning. Conversely, the validation data generator (validation_generator) has shuffle set to False to ensure consistency in validation performance evaluation.

5.3 Mobile Net V1 Architecture

5.3.1 Overview of The Architecture

Mobile Net, often referred to as MobileNetV1, is a groundbreaking family of convolutional neural networks designed to address the computational and memory constraints of mobile and embedded devices. Introduced by Google researchers in 2017, MobileNetV1 is celebrated for its exceptional efficiency while delivering robust performance in image classification tasks. At the heart of MobileNetV1's efficiency is the innovative use of depth wise separable convolutions, which significantly reduces the model's size and computational requirements. This architectural design is complemented by two crucial hyper parameters: the width multiplier and the resolution scale, allowing for customization of the model to match specific resource constraints. MobileNetV1 has found widespread application in mobile vision tasks, such as image classification, object detection, and semantic segmentation, and has served as a foundation for subsequent versions like MobileNetV2 and MobileNetV3, which further enhance the trade-off between model accuracy and efficiency. This pioneering neural network family has played a pivotal role in enabling deep learning on resource-constrained devices, empowering the development of lightweight yet powerful computer vision solutions for mobile and embedded systems [8]. Here are some key features and details about MobileNetV1:

1. **Efficiency**: MobileNetV1 is specifically designed to be computationally efficient, making it well-suited for deployment on mobile devices and embedded systems. It achieves this efficiency by using depthwise separable convolutions.

2. **Depthwise Separable Convolution**: MobileNetV1 uses a novel type of convolution operation called depthwise separable convolution. This operation splits the standard convolution into two separate layers: depthwise convolution and pointwise convolution. This significantly reduces the number of parameters and computations compared to traditional convolutions.

3. **Architecture**: MobileNetV1 consists of multiple layers, including depthwise separable convolution layers, followed by pointwise convolution layers. It typically ends with global average pooling and a fully connected layer for classification.

4. **Hyper parameter Tuning**: MobileNetV1 allows for hyperparameter tuning to balance the trade-off between model size and accuracy. You can choose different model variants by adjusting hyper parameters like the width multiplier and resolution scale.

https://doi.org/10.46335/IJIES.2023.8.10.2 **e-ISSN: 2456-3463**
Vol. 8, No. 10, 2023, PP. 55- 58

International Journal of Innovations in Engineering and Science, www.ijies.net

5. **Width Multiplier**: The width multiplier parameter can be adjusted to scale the model's width (number of channels) while keeping the depth (number of layers) constant. This parameter allows you to customize the model's size and computational requirements.

6. **Resolution Scale**: The resolution scale parameter allows you to resize the input images, which can also impact the model's size and computational demands. It is especially useful for adapting the model to the specific requirements of a mobile application

7. **Retrained Models**: Retrained MobileNetV1 models are available, which can be fine-tuned for specific tasks or used as feature extractors in transfer learning.

8. **Applications**: MobileNetV1 is commonly used in various mobile vision applications, including image classification, object detection, and semantic segmentation. It strikes a balance between accuracy and resource constraints, making it suitable for real-time inference on mobile devices.

9. **Successor Models**: MobileNetV1 has been succeeded by MobileNetV2 and MobileNetV3, which introduced improvements in terms of accuracy and efficiency. These subsequent versions build upon the foundations laid by MobileNetV1.

5.3.2 Architecture

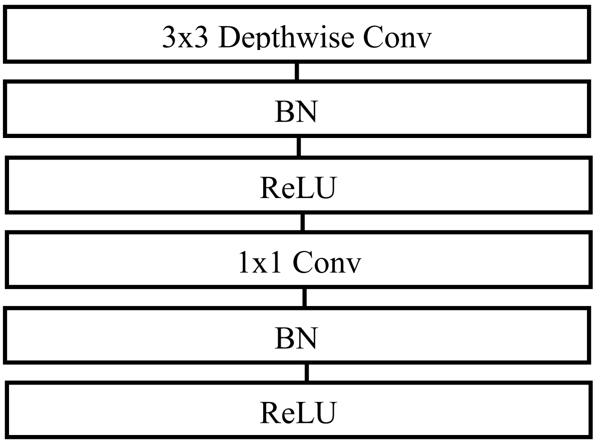

Figure 5.3.2.1: Depthwise Separable convolutions with Depthwise and Pointwise layers followed by batchnorm and ReLU. [8]

The above flow consists of a 3x3 depthwise separable convolution followed by batch normalization (BN) and rectified linear unit (ReLU) activation, and then a 1x1 pointwise convolution, again followed by BN and ReLU. The 3x3 depthwise convolution efficiently captures spatial dependencies in the input data by applying separate convolutions to each channel, reducing computational cost. Batch normalization normalizes the output, aiding convergence and generalization, and ReLU introduces non-linearity. The subsequent 1x1 convolution projects the depth of the feature space, allowing for a richer representation. This pattern is repeated throughout the MobileNet architecture, maintaining lightweight and efficient characteristics, except for the first layer which typically employs a regular convolution to capture broader patterns in the input data [8]. Below is the full architecture of mobileNetV1:

Table 5.3.2.1: Architecture of MobileNetV1 [8]

TYPE	STRIDE	KERNEL SHAPE	INPUT SIZE	OUTPUT SIZE
Convolution	2	3 × 3 × 3 × 32	224 × 224 × 3	112 × 112 × 32
Conv pw	1	1 × 1 × 32 × 64	112 × 112 × 32	112 × 112 × 64
Conv dw	2	3 × 3 × 64	112 × 112 × 64	56 × 56 × 64
Conv pw	1	1 × 1 × 64 × 128	56 × 56 × 64	56 × 56 × 128
Conv dw	1	3 × 3 × 128	56 × 56 × 128	56 × 56 × 128
Conv pw	1	1 × 1 × 128 × 128	56 × 56 × 128	56 × 56 × 128
Conv dw	2	3 × 3 × 128	56 × 56 × 128	56 × 56 × 128
Conv pw	1	1 × 1 × 128 × 256	56 × 56 × 128	28 × 28 × 256
Conv dw	1	3 × 3 × 256	28 × 28 × 256	28 × 28 × 256
Conv pw	1	1 × 1 × 256 × 256	28 × 28 × 256	28 × 28 × 256
Conv dw	2	3 × 3 × 256	28 × 28 × 256	14 × 14 × 256
Conv pw	1	1 × 1 × 256 × 512	14 × 14 × 256	14 × 14 × 512
Conv dw	1	3 × 3 × 512	14 × 14 × 512	14 × 14 × 512
Conv pw	1	1 × 1 × 512 × 512	14 × 14 × 512	14 × 14 × 512
Conv dw	1	3 × 3 × 512	14 × 14 × 512	14 × 14 × 512
Conv pw	1	1 × 1 × 512 × 512	14 × 14 × 512	14 × 14 × 512
Conv dw	1	3 × 3 × 512	14 × 14 × 512	14 × 14 × 512
Conv pw	1	1 × 1 × 512 × 512	14 × 14 × 512	14 × 14 × 512
Conv dw	1	3 × 3 × 512	14 × 14 × 512	14 × 14 × 512
Conv pw	1	1 × 1 × 512 × 512	14 × 14 × 512	14 × 14 × 512
Conv dw	1	3 × 3 × 512	14 × 14 × 512	14 × 14 × 512
Conv pw	1	1 × 1 × 512 × 512	14 × 14 × 512	14 × 14 × 512
Conv dw	2	3 × 3 × 512	14 × 14 × 512	7 × 7 × 512
Conv pw	1	1 × 1 × 512 × 1024	7 × 7 × 512	7 × 7 × 1024
Conv dw	2	3 × 3 × 1024	7 × 7 × 1024	7 × 7 × 1024

Conv pw	1	1 × 1 × 1024 × 1024	7 × 7 × 1024	7 × 7 × 1024
Avg Pool	1	Pool 7 × 7	7 × 7 × 1024	1 × 1 × 1024
FC	1	1024 x 1000	1 x 1 x 1024	1 x 1 x 1000
Softmax	1	Classifier	1 x 1 x 1000	-

5.4 Modification in MobileNet (New Architecture)

Table 5.4.1: Modified Architecture of MobileNetV1

TYPE	STRIDE	KERNEL SHAPE	INPUT SIZE	OUTPUT SIZE
Convolution	2	3 × 3 × 3 × 32	224 × 224 × 3	112 × 112 × 32
Conv pw	1	1 × 1 × 32 × 64	112 × 112 × 32	112 × 112 × 64
Conv dw	2	3 × 3 × 64	112 × 112 × 64	56 × 56 × 64
Conv pw	1	1 × 1 × 64 × 128	56 × 56 × 64	56 × 56 × 128
Conv dw	1	3 × 3 × 128	56 × 56 × 128	56 × 56 × 128
Conv pw	1	1 × 1 × 128 × 128	56 × 56 × 128	56 × 56 × 128
Conv dw	2	3 × 3 × 128	56 × 56 × 128	56 × 56 × 128
Conv pw	1	1 × 1 × 128 × 256	56 × 56 × 128	28 × 28 × 256
Conv dw	1	3 × 3 × 256	28 × 28 × 256	28 × 28 × 256
Conv pw	1	1 × 1 × 256 × 256	28 × 28 × 256	28 × 28 × 256
Conv dw	2	3 × 3 × 256	28 × 28 × 256	14 × 14 × 256
Conv pw	1	1 × 1 × 256 × 512	14 × 14 × 256	14 × 14 × 512
Conv dw	1	3 × 3 × 512	14 × 14 × 512	14 × 14 × 512
Conv pw	1	1 × 1 × 512 × 512	14 × 14 × 512	14 × 14 × 512
Conv dw	1	3 × 3 × 512	14 × 14 × 512	14 × 14 × 512
Conv pw	1	1 × 1 × 512 × 512	14 × 14 × 512	14 × 14 × 512
Conv dw	1	3 × 3 × 512	14 × 14 × 512	14 × 14 × 512
Conv pw	1	1 × 1 × 512 × 512	14 × 14 × 512	14 × 14 × 512
Conv dw	1	3 × 3 × 512	14 × 14 × 512	14 × 14 × 512
Conv pw	1	1 × 1 × 512 × 512	14 × 14 × 512	14 × 14 × 512
Conv dw	1	3 × 3 × 512	14 × 14 × 512	14 × 14 × 512
Conv pw	1	1 × 1 × 512 × 512	14 × 14 × 512	14 × 14 × 512
Conv dw	2	3 × 3 × 512	14 × 14 × 512	7 × 7 × 512
Conv pw	1	1 × 1 × 512 × 1024	7 × 7 × 512	7 × 7 × 1024
Conv dw	2	3 × 3 × 1024	7 × 7 × 1024	7 × 7 × 1024
Conv pw	1	1 × 1 × 1024 × 1024	7 × 7 × 1024	7 × 7 × 1024
Avg Pool	1	Pool 7 × 7	7 × 7 × 1024	1 × 1 × 1024
Flattern	-	-	1 × 1 × 1024	1024
Dense	-	-	1024	1
Activation	-	-	1	1

The architecture begins with a standard convolutional layer with a 3x3 kernel and 32 output channels, reducing the input image size from 224x224 to 112x112. Following this, there's a depthwise separable convolution, maintaining the spatial dimensions at 112x112. Then, a pointwise convolution is applied to expand the number of output channels to 64. Subsequently, another depthwise separable convolution with a stride of 2 reduces the spatial dimensions to 56x56. Pointwise convolutions increment the output channels to 128, and this pattern continues with alternating depthwise separable and pointwise convolutions.

This sequence of depthwise and pointwise convolutions effectively captures features and reduces spatial dimensions. The process culminates with global average pooling, reducing the spatial dimensions to 1x1. Following that, a flatten layer transforms the 1x1x1024 feature map into a 1D vector of size 1024. This vector is connected to a dense layer that serves as the final layer before classification, with 1024 input nodes and a single output node, typically used for binary classification. An activation layer applies a sigmoid function to the dense layer's output, producing the final prediction probability. This architecture is well-suited for image classification tasks, and the detailed table helps clarify the transformation of data at each stage of the network.

Example

Consider that an image of size 224x224x3 where 224 is the height and width of the image and 3 is the input channels (3 is for RGB). (The MobileNet architecture by default takes the image of size 224x244. So the images in dataset is resized to 244x244).

1. **Convolution Layer (Conv)**: The journey begins with a conventional convolutional layer. The input image, with its 3 color channels and dimensions of 244x244, undergoes a convolution operation using a set of 3x3x3x32 filters. This initial operation aims to extract various features from the image. With a stride of 2, the

https://doi.org/10.46335/IJIES.2023.8.10.2 e-ISSN: 2456-3463
Vol. 8, No. 10, 2023, PP. 55- 58

spatial dimensions are reduced, resulting in a 112x112x32 feature map.

2. **Depthwise Separable Convolution Layer (Conv dw)**: Following the first convolution, a depthwise separable convolution is applied. This layer comprises a depthwise convolution followed by a pointwise convolution. The input dimensions, 112x112x32, remain unchanged as the depthwise convolution maintains spatial size, while the pointwise convolution adjusts the number of channels. The aim here is to enhance feature extraction.

3. **Pointwise Convolution Layer (Conv pw)**: In the next step, a pointwise convolution is employed to increase the number of output channels from 32 to 64. This operation injects further complexity into the features extracted in the previous layers without altering the spatial dimensions, yielding a 112x112x64 feature map.

4. **Depthwise Separable Convolution Layer (Conv dw)**: Another depthwise separable convolution follows, but this time with a stride of 2. As a result, the spatial dimensions are reduced to 56x56 while preserving and improving the feature set within the 64 channels.

5. **Pointwise Convolution Layer (Conv pw)**: To further enrich the extracted features, a pointwise convolution is applied, increasing the number of output channels from

6. 64 to 128. This step increases the depth of features without impacting the spatial dimensions, resulting in a 56x56x128 feature map.

7. **Repeating Depthwise Separable and Pointwise Convolution Layers**: The architectural pattern of depthwise separable convolutions and pointwise convolutions persists, each with distinct kernel sizes and strides. Each depthwise separable convolution reduces spatial dimensions, and the subsequent pointwise convolution adjusts channel numbers. This sequential process is crucial for capturing intricate features within the data.

8. **Global Average Pooling (Avg Pool)**: At this stage, the network applies global average pooling, a critical step in feature reduction. This operation computes the average values across the spatial dimensions, effectively compressing the output into a 1x1x1024 feature map. The purpose of this step is to prepare the features for final classification.

9. **Flatten Layer (Flatten)**: The 1x1x1024 feature map is then flattened into a 1D vector comprising 1024 elements. This vector now serves as the input for the next layer.

10. **Dense Layer (Dense)**: The flattened vector enters a dense layer with 1024 input nodes and a single output node. This dense layer is commonly used for binary classification tasks. The output represents a numerical value, a crucial intermediary step towards making the final classification decision.

11. **Activation Layer (Activation)**: To convert the output from the dense layer into a probability, an activation layer applies a sigmoid function. This function maps the value to a range between 0 and 1, making it suitable for binary classification. The final output reflects the probability of the image belonging to a specific class.

5.5 Model Compilation and Training

5.6.5 Model Compilation:

The model compilation stage is crucial for defining the training objectives, optimizing strategy, and evaluation metrics tailored to the drowsiness detection task.

Loss Function (loss='binary_crossentropy'): Binary crossentropy is aptly chosen as the loss function, given the binary nature of the classification task. It quantifies the dissimilarity between the predicted probability distribution of eye states and the actual binary labels, providing a clear signal for the model to minimize the discrepancy.

Optimizer (optimizer='adam'): The Adam optimizer is employed due to its effectiveness in adjusting model weights efficiently. Its adaptive learning rate mechanism is beneficial for tasks with varying complexities, such as those encountered in real-time drowsiness detection scenarios.

Metrics (metrics=['accuracy']): Accuracy is a crucial metric for this application, as correctly identifying whether the driver's eyes are open or closed is paramount for accurate drowsiness detection. Monitoring accuracy during training provides insight into the model's ability to make correct predictions, which is directly relevant to the model's real-world performance.

5.6.6 Model Training:

The training process is designed to enable the model to learn the patterns indicative of open or closed eyes, leveraging a combination of data augmentation and transfer learning.

https://doi.org/10.46335/IJIES.2023.8.10.2 e-ISSN: 2456-3463
Vol. 8, No. 10, 2023, PP. 55- 58

- **Training Data (data_generator):** The **data_generator** is configured to dynamically augment the training dataset. Data augmentation is particularly beneficial for tasks like drowsiness detection, where variations in lighting conditions, head poses, or facial expressions can be present. Augmenting the dataset with variations in eye states enhances the model's ability to generalize to diverse scenarios.

- **Steps per Epoch:** The number of steps per epoch is determined by the total number of training samples divided by the batch size. Each step involves processing a batch of augmented images and updating the model's weights. In the context of drowsiness detection, this iterative learning process helps the model adapt to different driver conditions.

- **Epochs (epochs=8):** Training is performed over multiple epochs to allow the model to iteratively refine its parameters. This is crucial for capturing nuanced patterns associated with eye states and ensuring the model generalizes well to various scenarios encountered during driver monitoring.

- **Validation Data (validation_data=validation generator):** The **validation_generator** provides a separate set of images that the model has not seen during training. Evaluating the model on this validation set allows for assessing its generalization performance. For drowsiness detection, it ensures that the model can reliably identify open and closed eyes in new instances, crucial for real-world applicability.

5.6.7 Training Process:

The steps representing the flow of training of model from initial state to the state of saving history is given below:

1. **Initialization:**
   - The model's weights are initialized based on the architecture specified during model creation.
   - The Adam optimizer is initialized with its parameters, including learning rates and momentum coefficients.
2. **Data Augmentation:**
   - For each training batch, the data generator dynamically augments the images. This includes random rotations, shifts, shearing, zooming, and horizontal flips.
   - Data augmentation introduces variability into the training set, making the model more resilient to diverse real-world conditions.
3. **Forward Pass:**
   - Augmented images from the current batch are fed into the model for a forward pass.
   - The model processes each image through its layers, generating predictions for whether the eyes are open or closed.
4. **Loss Computation:**
   - The binary crossentropy loss is computed by comparing the model's predictions with the actual labels (ground truth) of eye states in the training batch.
   - The loss quantifies the difference between predicted probabilities and true labels, providing a measure of how well the model is performing on the current batch.
5. **Backward Pass (Backpropagation):**
   - Gradients of the loss with respect to the model's parameters (weights) are calculated during backpropagation.
   - These gradients represent the direction and magnitude of the adjustments needed to minimize the loss.
6. **Optimizer Update:**
   - The Adam optimizer utilizes the gradients to update the model's weights. The adaptive learning rate ensures efficient adjustments, especially in the presence of varying gradients across different parameters.
7. **Iterative Process:**
   - Steps 2-6 are repeated for each batch in the training set. The model processes multiple batches per epoch, and each batch contributes to the overall adjustment of the model's parameters.
   - This iterative process allows the model to learn progressively complex representations from the training data, capturing features associated with open and closed eyes.
8. **Epoch Progression:**
   - As the model goes through each batch in the training set, it refines its parameters to improve accuracy and reduce the loss.
   - The model continues this process for the specified number of epochs, iterating over the entire training dataset multiple times.
9. **Validation Evaluation:**
   - After completing each epoch, the model is evaluated on a separate validation dataset.
   - The validation set provides an unbiased assessment of the model's generalization performance, indicating how well it can predict eye states on unseen data.
10. **Training History:**
    - The training history, stored in the history variable, records metrics such as training and validation loss, training and validation accuracy, and any other specified metrics.
    - Analyzing the training history over epochs helps in understanding the model's learning dynamics and potential areas for improvement or adjustments in the

https://doi.org/10.46335/IJIES.2023.8.10.2 e-ISSN: 2456-3463
Vol. 8, No. 10, 2023, PP. 55- 58

training process.

5.6 Hardware Setup

5.6.1. Circuit Diagram

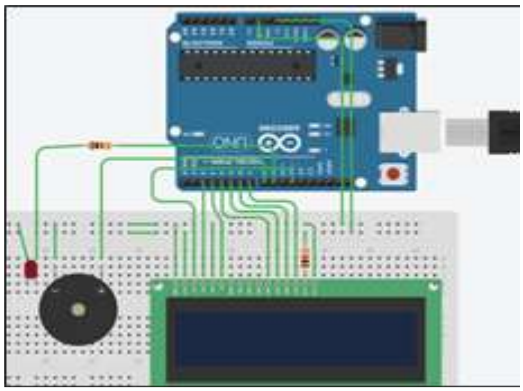

Fig 5.6.1.1: Alarm System Setup Using Arduino [9]

In the given circuit diagram, the ‘GND’ pin of LCD is connected to ‘GND’ pin of Arduino. ‘VCC’ pin of LCD is connected to ‘5V’ pin of Arduino. Similarly, backlight of LCD i.e. LED’s (used in LCD) cathode and anode is connected to ‘GND’ and ‘5V’ pin of Arduino respectively. ‘V0’ pin of LCD is connected to 11 no. pin of Arduino. It is used to adjust the brightness using pulse-width modulation (PWM) which is set 60. ‘RW’ pin of LCD is connected to ‘GND’ of Arduino since data will not be read from the LCD. ‘RS’ and ‘E’ pin of LCD is connected to 2 and 3 no. pin of Arduino respectively. ‘DB4’, ‘DB5’, ‘DB6’ and ‘DB7’ pin of LCD is connected to 4, 5, 6, and 7 no. pin of Arduino respectively. Buzzer is operated on 8 no. pin of Arduino. LED is operated on 9 no. pin of Arduino.

5.6.2. Code Implementation

In the loop function, the system continuously listens for incoming data on the serial port using Serial.available(). When data is received, the system reads the status and responds accordingly. If the received status is 'a', implying that the driver is drowsy and needs to wake up, the LCD displays a prompt to wake up, the buzzer is activated, and an LED is turned on for a brief duration, creating a noticeable alert. On the other hand, if the received status is 'b', indicating that the driver is alert and all is well, the LCD displays an "All Ok" message, and both the buzzer and LED are turned off after a short delay. The clearSerialBuffer() function is called to ensure any residual data in the serial buffer is cleared, preventing interference with subsequent communications. This loop continues to run, effectively monitoring the serial port for incoming instructions, allowing the system to respond in real-time to changes in the driver's alertness.

5.7 Integrating Trained Model and Hardware

To integrate the trained model and hardware setup, a python script is developed which will use the trained model to generate the input for the hardware setup and send it to the Arduino. Arduino, on the basis of input given will respond as mentioned earlier. The script utilizes the OpenCV library for face and eye detection, TensorFlow for loading a pre-trained deep learning model, and the Serial library for communication with an Arduino connected to the COM3 port.

The code begins by establishing a serial connection with the Arduino board. It then loads a pre-trained deep learning model for drowsiness detection, which is the pre-trained model. The script initializes cascade classifiers for face and eye detection using Haar cascades. The face detection is performed with lower parameters to enhance detection in less clear conditions. The webcam feed is captured using OpenCV's VideoCapture, and the program continuously processes frames from the webcam in a loop.

The script first converts each frame to grayscale to simplify processing. It then uses the face cascade classifier to detect faces in the frame. The largest detected face is selected, and its region of interest (ROI) is extracted. Within this ROI, the script uses the eye cascade classifier to detect eyes.

If eyes are detected, the region of the eyes is extracted and preprocessed for input into the pre-trained model. The pre-processing involves resizing the eye region to 224x224 pixels, normalizing pixel values to the range [0, 1], and expanding the dimensions to match the model's input shape. The pre-processed image is fed into the loaded model, and the predictions are obtained. If the prediction score for closed eyes is above a certain threshold (0.85 in this case), the script sends the status 'b' (indicating open eyes) to the Arduino via serial communication. If the prediction indicates closed eyes, a counter is incremented, and if the counter exceeds a certain threshold (3 in this case), the status 'a' (indicating closed eyes) is sent to the Arduino.

The script then updates the display frame with the detected status ('Open eyes' or 'Closed eyes') and visual cues. The frame is shown in an OpenCV window titled 'Drowsiness Detection'. The script exits when the 'q' key is pressed. To run this script, first connect the hardware setup to the system and load the code into the Arduino, which makes it active. Then run this python script.

https://doi.org/10.46335/IJIES.2023.8.10.2 e-ISSN: 2456-3463
Vol. 8, No. 10, 2023, PP. 55- 58

International Journal of Innovations in Engineering and Science, www.ijies.net

VI -RESULT

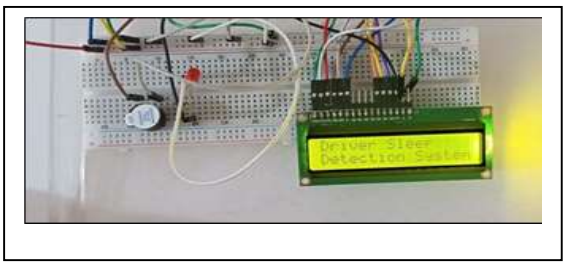

Figure 6.1: Initial Hardware

Figure 6.2: Open Eyes Detected

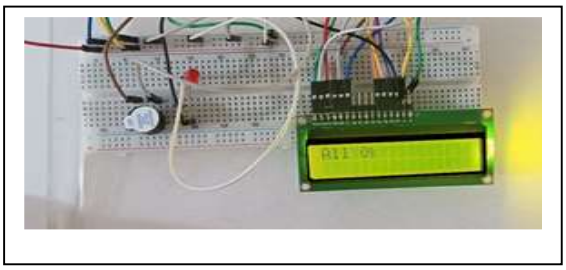

Figure 6.3: Hardware during Open Eyes Detected

Figure 6.4: Closed Eyes Detected

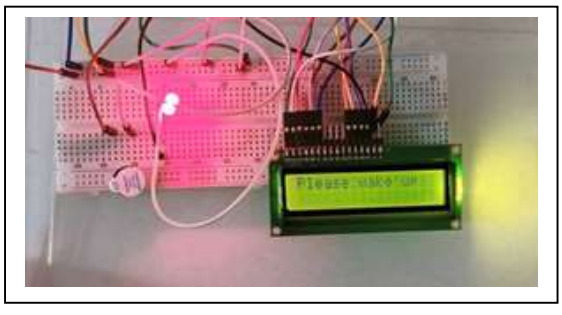

Figure 6.5: Hardware when Closed Eyes Detected

VII-CONCLUSION

In conclusion, the development and evaluation of the Driver Drowsiness Detection System mark a significant stride towards advancing road safety and prioritizing driver well-being. The comprehensive analysis of both hardware and software components, coupled with the exploration of functional and non-functional requirements, underscore the system's robustness and practicality.

The advantages offered by the system, including accident prevention, enhanced road safety, and real-time monitoring, position it as a valuable tool in mitigating the risks associated with drowsy driving. Moreover, the system's integration potential with other safety features and its adaptability to various vehicle types and transportation scenarios enhance its versatility.

Additionally, the emphasis on feasibility and economic viability, while deferring the use of night vision cameras to future iterations, reflects a thoughtful approach to project constraints and scalability.

As the system evolves, the outlined future scope introduces exciting possibilities, from advanced behavioural monitoring to customizable alert levels and integration with machine learning for predictive capabilities. These advancements position the Driver Drowsiness Detection System not only as a preventive measure against accidents but also as a proactive solution that evolves with the ever-changing landscape of driver safety.

In essence, the Driver Drowsiness Detection System, with its successful training outcomes and promising potential, stands as a testament to the commitment to road safety, ushering in a new era where technology plays a pivotal role in safeguarding lives on the road.

FUTURE SCOPE

1) Advanced Behavioral Monitoring:
Future iterations can extend beyond monitoring eyelid movement, incorporating advanced behavioral cues such as yawning, head nodding, and facial expressions. This comprehensive approach will allow the system to gauge drowsiness levels more accurately.

2) Customizable Alert Levels:
The system's future development could include customizable alert levels based on individual driver preferences and habits. This would enable a personalized approach, where drivers can set the system to provide warnings according to their unique drowsiness thresholds.

3) Time-Dependent Monitoring:
Time-dependent monitoring can be introduced, allowing the detection system to adapt its sensitivity based on the time of day. For instance, during peak drowsiness hours, the system

could be more vigilant and issue alerts at lower drowsiness levels, enhancing safety during high-risk periods.

4) Usage Patterns Analysis:
In future, a feature for analyzing usage patterns over time is possible to implement. Thus, the system will be able to identify specific hours or conditions when drowsiness levels are consistently high, providing valuable insights for both drivers and fleet managers.

5) Integration of Night Vision Cameras:
While not currently implemented due to cost constraints, the future could see the incorporation of night vision cameras. This enhancement would significantly improve the system's accuracy during low-light conditions, ensuring robust drowsiness detection regardless of the time of day.

6) Cloud Connectivity:
The overall project can consider integrating cloud connectivity for real-time data analysis and remote monitoring. This would enable centralized tracking of multiple vehicles, providing valuable data for research and fleet management.

7) Enhanced User Interface:
If more customizations are needed in the detection system alone, like the customizable audio buzzer, an intuitive, customizable and informative user interface can be developed, possibly incorporating feedback mechanisms and performance analytics. A user-friendly interface can enhance user engagement and encourage proactive drowsiness management.

8) Collaboration with Vehicle Manufacturers:
This takes a step further in the application and implementation of the overall Driver Drowsiness Detection System. Collaboration opportunities with vehicle manufacturers to integrate the drowsiness detection system as a built-in feature in new vehicle models, may contribute to widespread adoption.

These future enhancements aim to not only refine the detection system's accuracy and effectiveness, but also pave the way for broader applications and industry collaboration.

REFERENCES

[1] *Sundram Ojha, Syed Ali Asim, Ankita Agrawal, "Driver Drowsiness Detection", International Journal of Scientific Research in Engineering and Management, Volume 07, Issue 05, ISSN 2582-3930, Pg 2 – 4, May 2023.*

[2] *K Vijaychandra Reddy, Sanjana Gadalay, "Real-Time Fatigue Detection System using OpenCV and Deep Learning", International Research Journal of Engineering and Technology, Volume 08, Issue 11, ISSN 2395-0056, Pg 806 – 809, Nov 2021.*

[3] *Arun Sahayadhas, Kenneth Sundaraj and Murugappan Murugappan, "Detecting Driver Drowsiness Based on Sensors: A Review", Sensors Open Access Journal, volume 01, Issue 07, ISSN 1424-8220, Pg 05 – 11, Dec 2012.*

[4] *Jyotsna Gabhane, Dhanashri Dixit, Pranali Mankar, Ruchika Kamble, Sayantani Gupta, "Drowsiness Detection and Alert System: A Review", International Journal for Research in Applied Science & Engineering Technology, Volume 6, Issue IV, ISSN 2321-9653, Pg 237 – 238, April 2018.*

[5] *Manu B.N, "Facial Features Monitoring for Real Time Drowsiness Detection", 2016 12th International Conference on Innovations in Information Technology (IIT), Issue 20, Pg 79 – 81, Mar 2017.*

[6] *Amna Rahman "Real Time Drowsiness Detection using Eye Blink Monitoring", Department of Software Engineering Fatima Jinnah Women University 2015 National Software Engineering Conference (NSEC 2015).*

[7] *Eyes dataset: http://mrl.cs.vsb.cz/eyedataset*

[8] *Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), April 2017.*

[9] *Hardware circuit diagram on TinkerCad: https://www.tinkercad.com/things/7xZNDnzC1wN-driver-drowsiness-detection-system*