National College of Ireland

# Real Time Driver Drowsiness Detection Based on Deep Learning

MSc Research Project
Data Analytics


Shreyansh Gupta
Student ID: x21239347

School of Computing
National College of Ireland


Supervisor : Abid Yaqoob

**National College of Ireland**
**Project Submission Sheet**
**School of Computing**

| | |
|---|---|
| **Student Name:** | Shreyansh Gupta |
| **Student ID:** | x21239347 |
| **Programme:** | MSc Data Analytics |
| **Year:** | 2023 |
| **Module:** | MSc Research Project |
| **Supervisor:** | Abid Yaqoob |
| **Submission Due Date:** | 14/12//23 |
| **Project Title:** | Real Time Driver Drowsiness Detection based on Deep Learning |
| **Word Count:** | 7484 |
| **Page Count:** | 20 |

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

**ALL** internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

| | |
|---|---|
| **Signature:** | Shreyansh Gupta |
| **Date:** | 31st January 2024 |

## PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

| | |
|---|---|
| Attach a completed copy of this sheet to each project (including multiple copies). | ☐ |
| **Attach a Moodle submission receipt of the online project submission**, to each project (including multiple copies). | ☐ |
| **You must ensure that you retain a HARD COPY of the project**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. | ☐ |

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

| **Office Use Only** | |
|---|---|
| Signature: | |
| Date: | |
| Penalty Applied (if applicable): | |

# Real Time Driver Drowsiness Detection Based on Deep Learning

Shreyansh Gupta
x21239347

**Abstract**

Drowsiness refers to a state of feeling sleepy, sluggish, or fatigued. It is one of the biggest challenge faced by drivers all across the globe, contributing to a notable percentage of road accidents and fatalities. This research endeavors to develop and assess machine learning models tailored for eye detection and drowsiness prediction, targeting real-time applications. Motivated by the pivotal role of precise eye state detection in safety systems and human-computer interaction, Convolutional Neural Network (CNN) models, particularly leveraging the InceptionV3 architecture through transfer learning, were crafted to discern open and closed eyes. The model has achieved promising training accuracies ranging from 91.67% to 95.04%, and also on validation and test datasets, registering accuracies between 83.74% to 92.60%. Notably, varying convergence patterns led to early stopping at different epochs, highlighting challenges in achieving high generalization for practical deployment. These findings underscore the necessity for further research to enhance model reliability, generalization, and applicability in real-time scenarios, aiming to contribute to the development of robust safety systems reliant on accurate drowsiness detection.

**Keywords:** *Drowsiness Detection, Machine Learning Models, Convolutional Neural Networks (CNNs), Transfer Learning.*

## 1 Introduction

In the past few years, there has been a growing concern regarding the significant impact of drowsy drowsiness on road accidents, resulting in a noticeable number of fatalities and injuries on a global scale. The rise in the statistics associated with drowsy driving accidents justifies the need for advanced technological interventions to tranquilize this risk. A Real-Time driver drowsiness detection system emerges to be a promising solution to this problem, with the use of cutting-edge technologies to monitor and alert drivers when signs of drowsiness are detected.

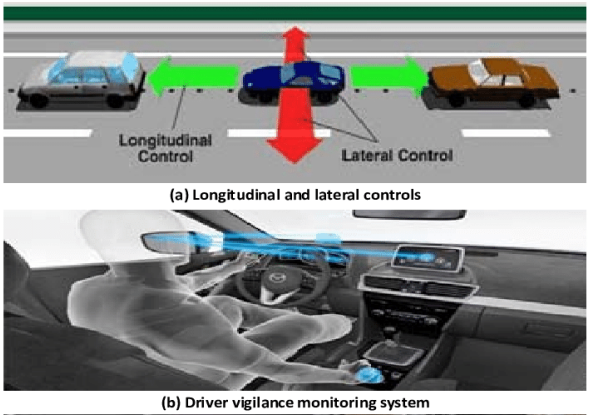
The E-Survey of Road Users' Attitudes (ESRA), [1] a worldwide study on driver behavior, found that 23.9% of Irish respondents had driven while tired and struggling to keep their eyes open at least once. This percentage surpasses the 20% average observed in other European nations. Also, Ireland's Road Safety Authority (RSA) [2] did a nationwide survey and found that 16% of 1,000 Irish drivers had admitted to falling asleep at the

[1] `https://www.esranet.eu/`

[2] Available at: `https://www.rsa.ie/`

wheel at least once. Drowsiness still remains a major problem as it is believed to be a leading cause of accidents and fatalities on Irish roads.

At present, Several driver drowsiness detection systems are available in modern vehicles. These systems often use various sensors and technologies to monitor driver behavior and alertness levels to prevent accidents caused by drowsy driving. One of the aspects of such systems is Vigilance Control. The best example of this system is the Ford Co-Pilot360™, it is an advanced suite of driver-assist technologies developed by Ford Motor Company. It's designed to enhance safety, convenience, and overall driving experience by incorporating several features that assist the driver in various scenarios. This system by Ford includes Automatic Emergency Braking (AEB), Adaptive Cruise Control with Stop-and-Go, etc. Although these systems do not directly hold a connection with drowsiness detection, they contribute in assisting the driver and prevention of road accidents. Figure 1 shows vigilance control on a vehicle. [3]




(a) Longitudinal and lateral controls

(b) Driver vigilance monitoring system

Figure 1: Vigilance Control

The MRL dataset was used in this research to train models while circumventing overfitting issues. Implementing various deep learning architectures—including Transfer learning and Convolutional Neural Networks. Finally, Real-time detection can be carried out with the help of a mounted camera in the car.

## 1.1 Research Questions

How effective is a fine-tuned convolutional neural network (CNN), utilizing the InceptionV3 pre-trained model, in accurately classifying eye states (open or closed) during the real-time driver drowsiness detection?

## 1.2 Research Objectives and Contributions

Adopting an innovative approach to this critical issue is of utmost importance, keeping in mind that existing improvements in drowsiness detection systems often limits to high-end and new vehicles, leaving older vehicle models unequipped with such safety measures.

[3]Available at:
https://www.researchgate.net/figure/Main-components-of-ADAS-namely-a-/
/longitudinal-control-and-lateral-control-b-driver_fig1_327192771

The challenge here at hand is to build an affordable, globally applicable technology capable of detecting these symptoms across diverse driving environments and vehicle types. Considering the vital role of behavioral signals in identifying drowsiness or fatigue, this research focuses on leveraging deep learning algorithms and computer vision techniques to craft a robust system proficient in precisely classifying drowsy drivers based on observable facial characteristics, primarily focusing on eye movements and closures. By integrating distinct facial features into the training of deep learning models, this approach aims to overcome the limitations of the previous systems that were developed for real-time drowsiness detection.

# 2 Related Works

In the domain of driver drowsiness detection, extensive research has been conducted employing various methodologies to enhance road safety and prevent accidents caused by driver fatigue. Notably, Convolutional Neural Network (CNN) based approaches and Employing image processing techniques, webcam monitoring, and innovative night vision camera utilization to detect signs of driver fatigue have gained traction, showcasing advancements in real-time detection with high accuracy and rapidity. These diverse methodologies underscore the ongoing efforts to address driver drowsiness detection from multiple angles, each with its strengths and limitations.

## 2.1 CNN-based Approaches

Hashemi et al. (2020) The presented paper utilized CNN for driver drowsiness detection represents a significant advancement, aiming for real-time application with high accuracy and rapidity. Introducing a novel dataset specifically designed for eye closure detection fills a critical gap in the availability of accurate eye datasets. However, a notable limitation lies in the reliance on a newly introduced dataset, which might require further validation and expansion to encompass diverse real-world scenarios and driver behaviors for robustness.

S. et al. (2022) Presented implementation of a drowsiness driving detection system using Raspberry Pi, leveraging CNN, showcases promising results in classifying drowsiness symptoms like blinking and yawning. The inclusion of a 4-layer convolution filter within the CNN architecture, coupled with training through the Adam optimization algorithm, yields a commendable classification accuracy rate ranging from 80% to 98%. However, despite the success in achieving high accuracy rates, limitations persist concerning the generalization of the system across diverse driving scenarios and driver behaviors

Dua et al. (2021) The proposed system for driver drowsiness detection represents a comprehensive approach utilizing multiple deep learning models to assess various driver-related features and behaviors. Employing models like AlexNet, VGG-FaceNet, FlowImageNet, and ResNet enables the consideration of diverse aspects such as environmental changes, facial characteristics, behavioral features, and hand gestures to detect drowsiness. The ensemble of these models, followed by a SoftMax classifier, demonstrates an encouraging accuracy of 85% in identifying drowsiness based on eye blinking, yawning, and nodding. However, despite its promising accuracy, potential limitations exist in real-world applicability concerning the system's performance under diverse driving conditions and the scalability of the approach for real-time deployment.

Chirra et al. (2019) The proposed framework leveraging deep learning for driver drowsiness detection, particularly focusing on eye state analysis, presents a promising advancement in enhancing road safety. Utilizing the Viola-Jones face detection algorithm for face and eye region extraction and a stacked deep CNN for feature extraction from dynamic key frames showcases a substantial accuracy improvement, reaching 96.42% compared to traditional CNN models. However, while the stacked deep CNN overcomes limitations of traditional CNN models, such as pose accuracy in regression, potential challenges might still exist in handling diverse driving conditions and individual variations. Further validation and refinement of the system under various environmental settings and driver behaviors could enhance its robustness.

R and Jacob (2022) In this paper, the implementation of a two-dimensional CNN-based classification model for driver drowsiness detection from facial images presents a promising stride in enhancing road safety. The comparative evaluation against transfer learning techniques like VGG-16 and ResNet-50 showcases superior performance of the proposed model, indicating its efficacy in accurately categorizing driver states into sleepy and non-sleepy classes. However, potential limitations might involve the model's performance in diverse environmental conditions, varying lighting, and its generalizability across different driver behaviors and populations.

Singh et al. (2021) The utilization of a mobile's front camera for detecting driver drowsiness represents a significant step toward enhancing road safety by leveraging portable and widely accessible technology. The focus on real-time detection and warning signal generation offers a proactive approach to mitigating accidents caused by driver fatigue. However, while this approach showcases promise, there might be limitations in achieving high accuracy and reliability due to potential challenges related to varying lighting conditions, camera angles, and device specifications across different mobile devices.

## 2.2 Haar Cascade / Real-Time Based Approaches

Fouzia et al. (2018) Proposed driver drowsiness detection system utilizing eye blink counts showcases promising results in accurately identifying driver fatigue. By continuously analyzing eye movements and triggering a vibrator alert when prolonged eye closure is detected, the system demonstrates effective drowsiness detection. Implemented in an OpenCV and Raspberry Pi environment with a single camera view, the system exhibits commendable performance. Limitations of the system, such as potential challenges in accurately detecting eye movements in varying lighting conditions or instances where the camera angle may affect the precision of eye tracking, thereby suggesting room for further enhancements to ensure robustness across diverse driving scenarios.

Chellappa et al. (2018) Developed driver drowsiness detection system utilizing a Raspbian camera and Raspberry Pi 3 presents a significant stride towards preventing potential highway accidents caused by driver fatigue. By employing image processing techniques and Haar Cascade Classifiers to monitor blink duration and calculate Eye Aspect Ratio (EAR), the system effectively assesses the driver's level of alertness. However, this system's efficacy might be contingent on various factors such as environmental lighting conditions, camera positioning, and the individual variability in blinking patterns, which could influence the accuracy of drowsiness detection.

Kulkarni et al. (2017) The system, as presented in the paper, effectively utilizes image processing algorithms to monitor driver behavior and aid in vehicle control. By employ-

ing Raspberry Pi and a webcam, it demonstrates the potential for real-time monitoring and alert generation to ensure driver vigilance. However, certain limitations may exist, such as potential challenges in accurately capturing images under varying environmental conditions or limitations in the responsiveness of the system in high-speed scenarios

K et al. (2017) Drowsy Detection and Rescue System devised for night-time driving presents an innovative approach in addressing driver fatigue. The utilization of an infra-red night vision camera focused on the driver's face to monitor eye movements significantly contributes to detecting signs of fatigue. When fatigue is identified through closed eyes detected across consecutive frames, the system promptly issues an alarm to alert the driver. Furthermore, as a safety measure, the system autonomously parks the vehicle to the left side of the street upon detecting the driver's drowsiness. However, despite its novel approach and functionalities, this system might encounter limitations related to the accuracy of detecting fatigue solely based on eye closure.

Maior et al. (2018) The presented model for drowsiness detection through video stream analysis offers a non-intrusive and accessible approach for monitoring operator alertness in critical work environments. Utilizing computer vision and machine learning techniques, the system demonstrates the ability to detect drowsiness in real-time using a standard web camera, without the need for specialized biological sensors. By employing automatic face detection and Eye Aspect Ratio evaluation followed by Support Vector Machines classification, the system can promptly alert operators when signs of drowsiness are detected, potentially mitigating the risk of human error and reducing accidents in safety-critical workplaces. However, limitations may arise in the system's accuracy under varying lighting conditions, diverse facial expressions, and individual variability in alertness patterns.

In conclusion, This research addresses the prevalent limitations mentioned various papers on driver drowsiness detection systems by focusing on enhancing robustness and adaptability across diverse real-world driving scenarios. Leveraging the MRL eye dataset, our approach captures a wide spectrum of eye states, lighting conditions, facial characteristics, and driver behaviors, aiming to develop a more comprehensive model for real-time driver drowsiness detection. To mitigate challenges related to varying conditions, our strategy involves a robust preprocessing pipeline incorporating image augmentation and normalization techniques. Additionally, employing the InceptionV3 CNN architecture aids in improving feature extraction and classification accuracy, particularly in detecting diverse eye states under varying environmental and behavioral contexts.

# 3 Methodology

This project demonstrates a comprehensive process for building an eye state detection system using machine learning techniques and computer vision. Models that are based on deep learning and neural networks have shown promising results in such type of projects. This type of systems are usually based on the a person's physical signals such as eye movements. Several other systems that use behavioural psychological methods can result in a number of errors, and thus are not usually preferred over deep learning methods.

The utilization of the CRISP-DM (Cross-Industry Standard Process for Data Mining) [4] framework as a guiding methodology is found to be one of the best and widely-used analytics model. Figure 2 shows the CRISP-DM process for the research approach.

[4]Source: `https://datarundown.com/data-science-life-cycle/`

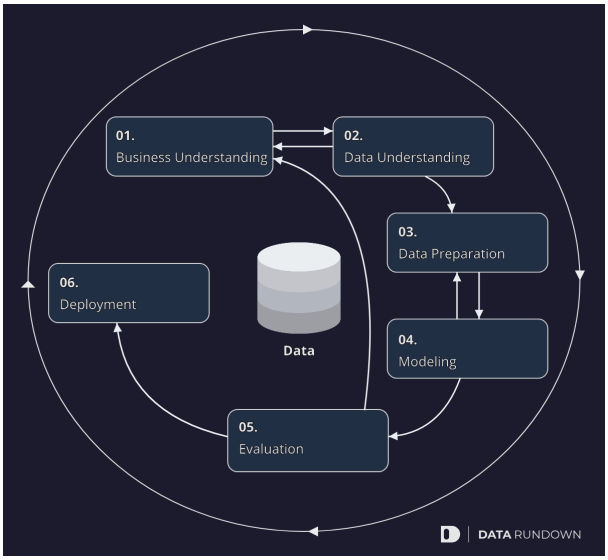



Figure 2: CRISP-DM

The CRISP-DM methodology is widely recognized as a comprehensive framework that outlines a structured series of steps that are crucial for achieving success in data mining and machine learning projects. This framework provides a clear and well-defined sequence of actions that need to be undertaken in order to effectively carry out such projects. The methodology discussed in this paper consists of five main phases: Project Understanding, Data Gathering, Data Preparation, Modeling, and Evaluation. During the study, similar steps were taken which will be discussed further in this section.

## 3.1 Research Overview

This project aims to reduce cases of vehicle crashes caused as a result of drowsiness on the side of vehicle drivers. As mentioned in Section 1, statistics from the government confirm that yearly thousands of accidents across the transportation sector are all due to driver fatigue, pointing its way substantial evidence that it is an issue at hand. The suggested development of an automated system of real-time detection of driver drowsiness makes it possible to prevent the problem through alerts prompting preventive actions before fatigue ensues causes collisions. The computer vision solution suggests non-intrusive monitor for drivers since a camera mounted facing towards the driver on the dashboard of the vehicle takes the signals used in detecting drowsiness by monitoring face features such as eyelid closure.

## 3.2 Data Gathering

The second major step of the CRSIP-DM methodology is understanding the data and its nature. The MRL eye dataset [5] was utilised in this research, this dataset is available publicly; although the main purpose of the dataset may vary, it is a good choice when it comes to this research which revolves around the physical movement of eyes. The mentioned dataset contains about 84k images belonging to 37 different people; 33 men

[5]Available at: `http://mrl.cs.vsb.cz/eyedataset`

and 4 women. Figure 3 given below shows all the properties and nature of the dataset. For example the 5th index defines whether the eyes are open or closed ie. 0 defines closed eyes and 1 defines open eyes.

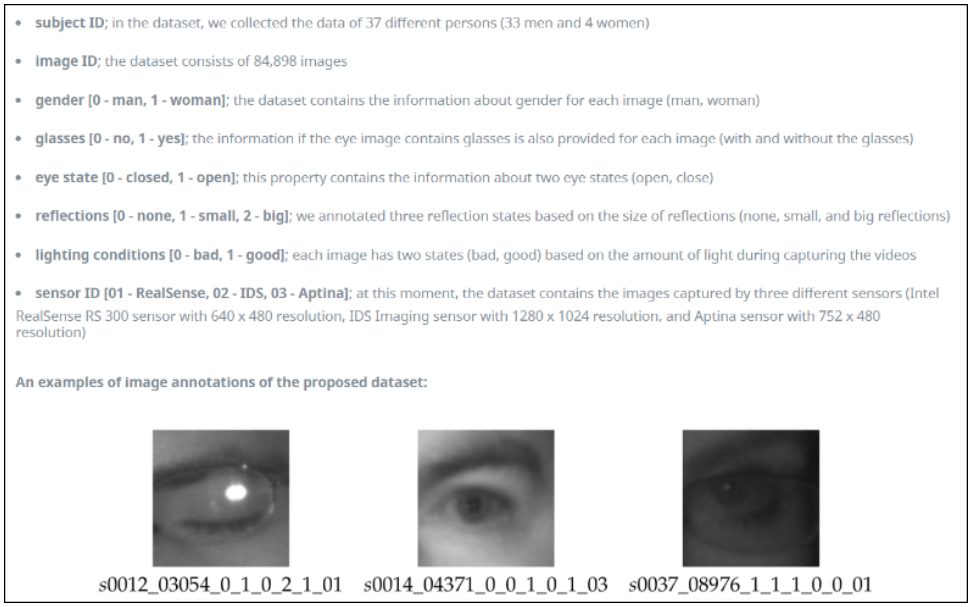



Figure 3: Naming Conventions of The Dataset

## 3.3 Data Pre-Processing

Data preparation or Data pre-processing is considered to be the most crucial step while following the CRISP-DM methodology. To begin with, the data was split with the help of *'shutil'* library into three sets; Training, Validation and Test sets. The split is crucial for training machine learning models and assessing their performance. The split ensures that the model is trained on a portion of the data while keeping another portion separate for validation purposes.

The split is performed using the *validation_split* parameter within the ImageDataGenerator class. Two ImageDataGenerator instances are created: *train_datagen and test_datagen*. *train_datagen* is used for generating augmented training data and has a parameter *validation_split=0.2*, indicating a 20% split for validation within the training dataset.

**Explanation of the split :**

- **Training Set (80%)**: The train_data generator is responsible for supplying augmented images to train the model. It contains 80% of the original dataset images based on the validation_split=0.2 parameter.
- **Validation Set (20%)**:

  The validation_data generator uses a subset (20% of the original dataset) for validation during model training. These images are not used for training the model but are essential to assess how well the model generalizes to unseen data.

After the separation of data into respective sets, the training data is then undergone augmentation to expand the total number of images by an order of magnitude.

The *ImageDataGenerator* class from the Keras deep learning library was used to automatically augment the training data on-the-fly during model training. This saves disk space compared to permanently transforming images. The *ImageDataGenerator* performs transformation like rotations up to 20 degrees, shearing, zooming between 0.8 to 1.2 times their original size, horizontal & vertical shifts up to 20% of the width/height. This diversifies the data to prevent over-fitting and improve generalization capacity. In total, over 5 augmented variants were generated per original training image. So the effective dataset size for training increased by 5-6x. A sample of the pre-processed Training dataset images are shown in Figure 4

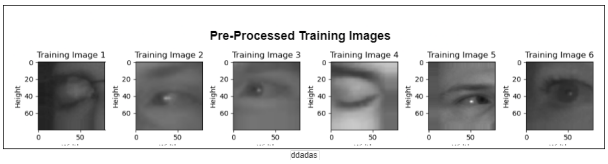



Figure 4: Pre-processed Training Images

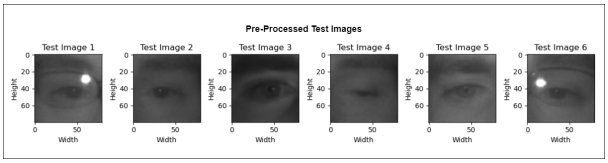



Figure 5: Pre-processed Test Images

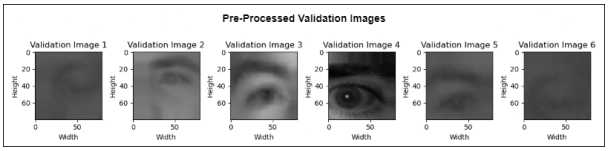



Figure 6: Pre-processed Validation Images

The images are then resized to 80x80 pixels and pixel intensities scaled from 0-255 to 0-1 for computational efficiency, since deep learning models requires large amount of training data, such pre-processing gives an edge for learning robust eye classifiers from limited data. Carefully labelling of open and closed eye images ensures low label noise. The pre-processed data was then visualized with the help of *'matplotlib'* library as can be seen in Figure 4 Figure 6 and Figure 5 respectively.

## 3.4 Modelling

In the CRISP-DM (Cross-Industry Standard Process for Data Mining) methodology, the modeling phase is a significant stage. This phase involves the actual construction of a predictive or descriptive model based on insights gained during the previous phases.

### 3.4.1 Convolutional Neural Networks (CNNs)

The CNN architecture Kim (2017) is defined as specialized neural networks designed for processing and analyzing visual data, in this case images. These neural networks consist of convolutional layers that apply filters to input images, capturing spatial hierarchies of features. CNNs possesses the capacity to automatically learn how to extract meaningful patterns, textures and hierarchical representations. Simple features such as edges and textures can be recognised in early layers while complex layers such as shapes and objects in deeper layers.

### 3.4.2 Transfer Learning

The model harnesses transfer learning Ali et al. (2023)by utilizing the pre-trained InceptionV3 [6]. model, based on CNN principles which automatically learns and extracts hierarchical representations from images. Pre-training on ImageNet provides InceptionV3 with the capability to recognize and extract diverse visual features from images.

The InceptionV3 architecture contains convolutional layers that perform feature extraction by applying various filter sizes within its inception modules, this architecture includes pooling layers and fully connected layers internally. Custom dense (fully connected) layers are added on top of the InceptionV3 base layers to perform eye state classification. These added layers take the flattened output from the preceding layers and perform classification based on the learned features. The layers of this pre-trained model are frozen during the model-building process which prevents their weights from being updated during training, preserving the learned representations from ImageNet. Only the added custom layers on top of the base are trained with the eye state dataset.

## 3.5 Evaluation

The evaluation of the aforementioned research project will encompass several crucial aspects to ascertain the performance and efficacy of the developed detection system. The evaluation will involve rigorous testing of the trained model using distinct performance metrics such as accuracy which will quantify the model's ability to accurately classify eye states (open or closed) based on unseen data, thereby assessing its overall predictive capability and in order to gauge the system's robustness and generalization capacity, extensive cross-validation techniques will be employed, ensuring the model performs consistently across diverse subsets of the dataset. In the end, real-time simulations or live testing within controlled environments emulating driving scenarios will be conducted to assess the system's practical utility and responsiveness in detecting driver drowsiness accurately. The evaluation process will aim to validate the model's performance, reliability, and feasibility as an effective tool in preventing vehicular accidents due to driver drowsiness.

# 4 Design Specification

In order to build a robust model, a well established architecture that captures each and every aspect of the model is designed. The Figure 7 given below shows the whole design specification that was utilized in order to carry out the research objectives.

[6]Available at: `https://keras.io/api/applications/inceptionv3/`

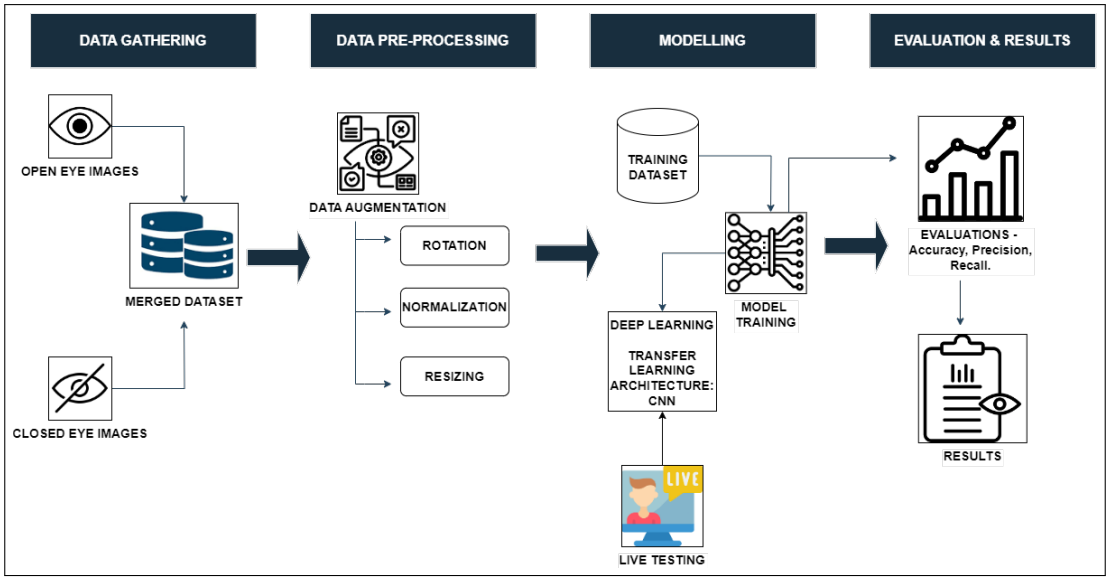



Figure 7: Flow of the Model

The system as can be seen in the Figure 7 functions by preparing the directories to organize image data for open eyes and closed eyes.

## 4.1 Data Specification

The data was extracted from the MRL website and then all the closed eye and opened eye images of 37 different people were separated with the help of naming conventions and were stored in *Closed_eye and Open_eye* directories. Both the Closed eye and open eye directories under the train dataset had 40k+ images respectively; from which, around 10% of the images were manually transferred to test dataset.

The images from here on are ready to be converted to a numpy array which can be further utilized by the CNN architecture. But before this, the images are augmented using *ImageDataGenerator* class from TensorFlow's Keras API. The data augmentation parameters involves : *rotation_range, shear_range, zoom_range, width_shift_range, height_shift_range* and more. These parameters define the range and types of transformations applied to augment the images during training. These transformations include rotation which was set to 0.2 meaning that random rotations will be applied within the range of -0.2 to 0.2 times the maximum rotation angle (in degrees), zooming and shifting both horizontally and vertically, thereby artificially increasing the diversity of the training dataset.

### 4.1.1 Normalization

Normalization or Rescaling involves transforming the pixel values of images to a specific range to ensure uniformity and aid in model convergence during training. By rescaling the pixel values within a certain range, typically [0, 1] or [-1, 1], the neural network can effectively learn patterns without being biased towards higher or lower values in the input data. In this case, The rescale=1./255 parameter in the *ImageDataGenerator* instance is used to perform rescaling of pixel values in the range [0, 1]. Dividing by 255 scales down the pixel values (which range from 0 to 255 in an 8-bit color image) to fall within the [0, 1] range. When the generator reads images during training or validation,

each pixel value is divided by 255. For instance, if an original pixel value is 127 in the image, after rescaling, it becomes 127 / 255 = 0.498, effectively transforming it to a value between 0 and 1. Normalizing pixel values helps in preventing the model from learning noise or unnecessary patterns due to varying pixel ranges, in simple terms it helps avoid overfitting.

## 4.2 Model Development

After the pre-processing of the data the next step is building a deep learning model for eye state classification. This deep learning model employs transfer learning by utilizing a pre-trained InceptionV3 as a base model and then custom layers were added for better eye state classification. This InceptionV3 was initially trained on ImageNet dataset, which is a large-scale dataset consisting of millions of labelled images which enables the neural network to learn high-level features from a diverse set of images. The model is trained on a number of epochs as it improves the performance of the model and it takes quite some time to train the model. Once the model is trained, it is saved so that it can be used for deploying real time detection. After saving the model, the next step is evaluating the performance. A few metrics that evaluate the performance are Accuracy, Precision, recall and F-1 score.

With the evaluations in hand, the final stage is deployment of real time detection and obtaining the results. Haar cascade along with Convolutional neural network was used to during the real time capturing of image to detect faces and objects. Accuracy is considered to be the major evaluation metrics along side the actual detection of open and closed eyes via the web cam.

## 4.3 Real-Time Implementation

Real-time implementation utilizes computer vision libraries and a pre-trained deep learning model to implement real-time driver drowsiness detection through a webcam feed. It begins by loading the necessary libraries, including OpenCV for image processing, TensorFlow for the deep learning model, and Pygame for sound alerts. The Haar cascades for face and eye detection are employed to identify facial regions and extract eyes from the video frames captured by the webcam.

The built model continuously captures video frames and processes each frame in a loop. It detects faces and eyes in the frame, then isolates and preprocesses the eye region for classification by the loaded deep learning model. Based on the model's predictions, it determines whether the eyes are open or closed. This determination triggers corresponding actions: incrementing or decrementing a 'Score' variable and displaying eye state information ('open' or 'closed') on the video frame.

A scoring system keeps track of eye state changes and triggers an alert sound when the eyes are predicted to be closed for a specified duration, indicating potential drowsiness. The real-time visual and scoring feedback on the frame assists in monitoring the user's eye status, and the sound alert provides a warning in case of prolonged eye closure, potentially aiding in preventing drowsiness-related incidents while driving or in other situations demanding alertness.

# 5 Implementation

This section of the research will describe how the whole design and model of real time driver drowsiness detection system was implemented, However as most of the steps of building this system were already mentioned in previous sections, this section shall delve deeper into the final stage of the whole implementation while also capturing the essence of beginning stages.As mentioned previously, The *MRL eye dataset* was utilised for conducting this research; Intially the dataset was not made for building such model but it proved out to be one of the best dataset for such researches as it holds the key elements which plays an important role in such studies.

The implementation process involves several steps as described in the design specification. Starting with Data collection and pre-processing to the final real time working of the built model.

## 5.1 Tools and Technologies

**Programming Languages:**

- **Python:** Python is chosen due to its versatility and ease of use, making it an ideal language for machine learning and AI-related tasks.

**Libraries/Frameworks:**

- **OpenCV (Open Source Computer Vision Library):** OpenCV is a powerful library used extensively for computer vision tasks. It provides functionalities for image and video analysis, including object detection, facial recognition, and image manipulation.
- **NumPy:** NumPy is a fundamental library for numerical computations in Python. It offers support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions for operating on these arrays.
- **TensorFlow and Keras:** TensorFlow is a popular open-source framework for machine learning and deep learning tasks.

  Keras, integrated within TensorFlow, offers a high-level neural networks API, simplifying the process of building and training neural networks.
- **Pygame:** Pygame is utilized for audio alerting purposes. Although primarily used for game development, Pygame's audio functionality suits alerting or notification requirements in applications beyond gaming.

## 5.2 Deep Learning Model

In order to forward, a deep learning model is compiled for eye state classification. In this case, the pre-trained InceptionV3 model, which has been trained on a large dataset like ImageNet for general image recognition, is used as a starting point for a new task—eye state classification. The focus here is on using the convolutional base of InceptionV3, a powerful feature extractor due to its numerous convolutional and pooling layers. After removing the top layers, customized layers were added. These additional layers are responsible for learning and making predictions regarding whether the eyes in an image are open or closed.

### 5.2.1 Model 1

**Base Model Initialization**

- **Custom Dense Layers (64 units, ReLU activation)** Addition of Dense layers helps in learning higher-level abstractions from the features extracted by the convolutional base. ReLU (Rectified Linear Unit) activation is chosen as it introduces non-linearity, allowing the model to learn complex relationships within the data.
- **Dropout Layer (0.5 dropout rate)** Dropout is applied for regularization, preventing overfitting by randomly dropping 50% of the neurons during training, which encourages the network to learn more robust features.
- **Final Dense Layer (2 units, Softmax activation)** oftmax activation is suitable for multi-class classification tasks, producing probabilities for the 'open' and 'closed' eye states. 2 units correspond to the number of classes for eye state classification (open or closed).

```
bmodel = InceptionV3(include_top=False, weights='imagenet',
input_tensor=Input(shape=(80,80,3)))
hmodel = bmodel.output
hmodel = Flatten()(hmodel)
hmodel = Dense(64, activation='relu')(hmodel)
hmodel = Dropout(0.5)(hmodel)
hmodel = Dense(2,activation= 'softmax')(hmodel)
```

Listing 1: Base Model Initialization

**Freezing Base Layer**

Pre-trained layers from InceptionV3 are set as non-trainable to retain the learned representations and prevent significant modification of these weights, also freezing these layers allows the model to focus on learning task-specific features while leveraging the general features learned from ImageNet.

```

bmodel = InceptionV3(include_top=False, weights='imagenet',
input_tensor=Input(shape=(80,80,3)))
for layer in bmodel.layers:
    layer.trainable = False
```

Listing 2: Freezing base layer

**Compilation of the Model:**

The model, comprising the InceptionV3 base and the added custom layers, is then compiled:

- **Optimizer (Adam)**: Adam optimizer [7] is chosen to update the model's parameters during training. It's known for its efficiency and adaptability in handling various datasets and models.

[7] Available at: `https://keras.io/api/optimizers/adam/`

- **Loss Function (Categorical Cross-Entropy)**: Categorical cross-entropy is selected as the loss function to measure the dissimilarity between the predicted eye states and the actual labels. This loss function is common in multi-class classification tasks.

```
model.compile(optimizer='Adam', loss='
   categorical_crossentropy',
metrics=['accuracy'])
```

Listing 3: Model Compilation

### 5.2.2 Model 2

In Model 2, the architecture is extended with two additional dense layers compared to Model 1, aiming to enhance the neural network's capacity for learning intricate representations. After the InceptionV3 base, a Flatten layer is employed to transform the 3D feature maps into a 1D vector. Subsequently, two densely connected layers are introduced, consisting of 128 and 64 neurons, respectively, both utilizing Rectified Linear Unit (ReLU) activation functions. To mitigate overfitting and improve generalization, dropout layers with a dropout rate of 0.5 are strategically placed after each of these additional dense layers. Finally, the output layer, akin to Model 1, comprises 2 units with a softmax activation function, facilitating binary classification (open or closed eyes). Notably, the model's training duration extends to 20 epochs, and the early stopping patience parameter is adjusted to 14, allowing the training process to continue for a longer period while monitoring performance improvements, thereby adapting to potential longer convergence times or complex learning patterns. The Listing 4 below shows the code snippet of model 2.

```
head_model2 = base_model2.output
head_model2 = Flatten()(head_model2)
head_model2 = Dense(128, activation='relu')(head_model2)
head_model2 = Dropout(0.5)(head_model2)
head_model2 = Dense(64, activation='relu')(head_model2)
head_model2 = Dropout(0.5)(head_model2)
head_model2 = Dense(2, activation='softmax')(head_model2)

earlystop = EarlyStopping(monitor = 'val_loss',
patience=14, verbose= 3, restore_best_weights=True)
```

Listing 4: Model 2

### 5.2.3 Model 3

The model is built on the same base architecture but exhibit difference in design complexity and layer configurations. It is more intricate, encompassing a sequence of 5 densely connected layers with increasing units—128, 64, 32, 16, and 8—interspersed with dropout layers before the final classification layer. This increased layer depth and complexity potentially allows for the extraction of more intricate and nuanced features from the input data but similar to previous models it freezes the pre-trained InceptionV3 base layers, enabling only the custom-added layers to be trained. This model was built for handling

larger and more intricate datasets, capturing finer details that were not captured in the previous models for improved classification accuracy. The Listing 5 below shows the code snippet of model 3.

```
base_model3 = InceptionV3(include_top=False,
weights='imagenet', input_tensor=Input(shape=(80, 80, 3)))

head_model3 = base_model3.output
head_model3 = Flatten()(head_model3)
head_model3 = Dense(128, activation='relu')(head_model3)
head_model3 = Dropout(0.5)(head_model3)
head_model3 = Dense(64, activation='relu')(head_model3)
head_model3 = Dropout(0.5)(head_model3)
head_model3 = Dense(32, activation='relu')(head_model3)# 3rd
   dense layer
head_model3 = Dropout(0.5)(head_model3)
head_model3 = Dense(16, activation='relu')(head_model3)# 4th
   dense layer
head_model3 = Dropout(0.5)(head_model3)
head_model3 = Dense(8, activation='relu')(head_model3)# 5th
   dense layer
head_model3 = Dropout(0.5)(head_model3)
head_model3 = Dense(2, activation='softmax')(head_model3)

model3 = Model(inputs=base_model3.input, outputs=head_model3)

for layer in base_model3.layers:
    layer.trainable = False
```

Listing 5: Model 3

# 6 Experiment Results & Evaluations

The study so far embarked on the task of eye state detection—discerning between open and closed eyes—utilizing machine learning techniques. The research aimed to develop models capable of identifying these states accurately. The experimental setup involved the creation and evaluation of models based on the InceptionV3 architecture. Through meticulous training and testing procedures on the proposed MRL eye datasets, the models were assessed for their efficacy in distinguishing eye states. Subsequently, real-time testing was conducted to validate the applicability of the models in practical scenarios. This section delves into the performance of these models, shedding light on strengths and limitations and also check the potential it holds in real-world scenarios by testing the model; feeding live images using the webcam of the system and noting the results it produces.

The evaluation metrics for this study is based on the categorical accuracy achieved and pateince parameter for different deep learning models and how well these models detect open and closed eyes in real time.

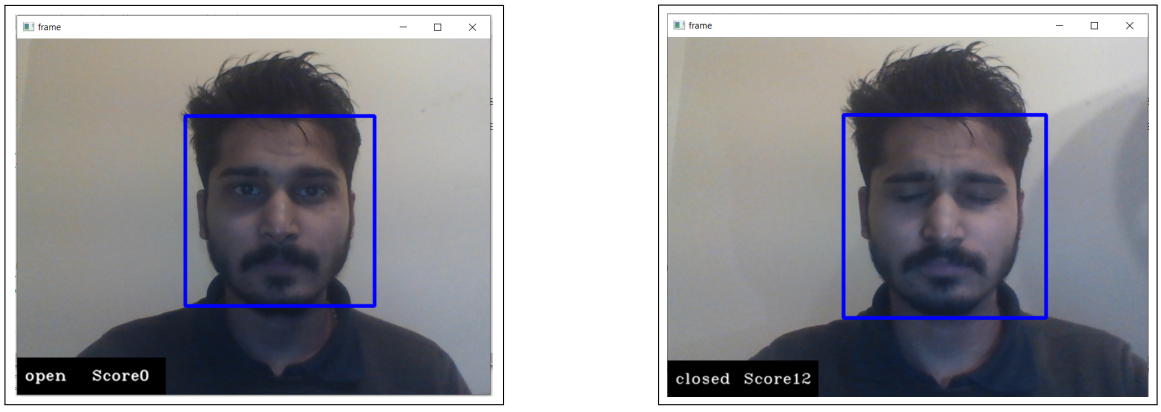
## 6.1 Exp. 1 (Base Model)

Experiment 1 is a deep learning model with a custom classification head that is added to the model consisting of a Flatten layer followed by a Dense layer with ReLU activation and a Dropout layer to prevent overfitting. The final Dense layer is with a softmax activation function that outputs probabilities for two classes. The model is then compiled using *adam* optimizer and categorical cross-entropy loss function. The training process employs callbacks such as ModelCheckpoint, EarlyStopping, and ReduceLROnPlateau for model saving, early stopping to prevent overfitting, and dynamically adjusting the learning rate, respectively. Under EarlyStopping, the patience parameter was set to 7 and it was run for 10 epochs.

The discrepancy between training and test accuracies (95.04% vs. 86.04%) shows a difference of 11% which suggests that the model might be overfitting the training data to some extent.

### 6.1.1 Real time testing

The model's real-time performance in discerning between open and closed eyes exhibited promising results as during the testing phase using a webcam feed, the model efficiently recognized open eyes, maintaining a near-zero score, indicative of its accurate identification. On the other hand, as soon as the eyes were detected as closed, the model's scoring mechanism gradually incremented, showing its responsiveness in detecting this state change. The recorded snapshots during the test session vividly depict instances of both open and closed eyes.




(a) Test 1: Open eyes

(b) Test 1: Closed eyes

Figure 8: Comparison of Test 1

What notably underscored the model's success was its discernment of closed eyes, triggering an alarm when the score surpassed a predefined threshold. This alarm acted as a real-time indicator, corroborating the model's accurate predictions and affirming its proficiency in distinguishing between open and closed eyes.

## 6.2 Exp. 2 (Additional Dense layers [128 and 64 units])

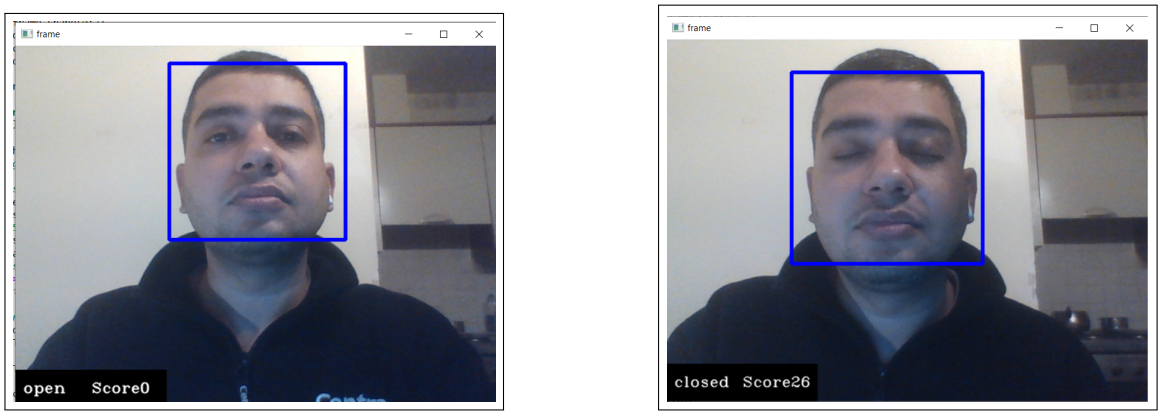
In order to increase the complexity of the model two extra Dense layers, one with 128 units and another with 64 units followed by Dropout layers were added. Each Dense layer

introduces more parameters and computation, allowing the network to learn more sophisticated representations from the data. The patience parameter within the EarlyStopping configuration underwent a modification; previously set at 7, it was extended to 14 in this updated model configuration. And finally the model was run for 20 epochs.

While the current model has a higher number of training epochs due to the increased patience in early stopping, it does not seem to outperform the previous model in terms of training and test accuracies (94.81% and 83.92% respectively. The current model shows slightly higher losses and slightly lower accuracies across training, validation, and test sets, suggesting it might not generalize as effectively as the previous model did.

### 6.2.1 Real Time Testing

Testing the model with a different face to see how well the model generalizes to new faces and different lightning conditions. As observed, The model exhibited a preference for higher light intensity to achieve optimal focus on the individual's face, indicating its sensitivity to lighting variations.[8].




(a) Test 2 : Open eyes

(b) Test 2 : Closed eyes

Figure 9: Comparison of Test 2

Subtle fluctuations in the model's score were observed during the testing process. These fluctuations could stem from inherent complexities in facial features or diverse facial expressions, underscoring the challenges in achieving absolute stability across various faces. This underscores the need for continued refinement or augmentation strategies to enhance the model's robustness and stability when confronted with diverse facial characteristics.

## 6.3 Exp. 3 (Five Dense Layers And Patience Constant)

The third experiment aim to provide the model with additional capacity to learn intricate patterns while incorporating dropout for regularization to mitigate overfitting. This model incorporates additional Dense layers with increasing sizes of 128, 64, 32, 16, and 8 units, respectively, followed by Dropout layers (with a dropout rate of 0.5) after each Dense layer.

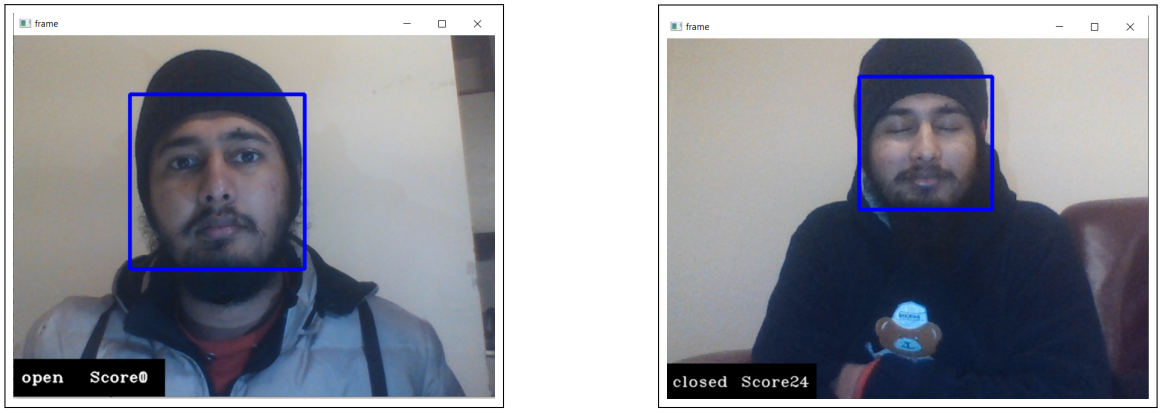
[8]Note: Proper consent was obtained from the person for the experiment

The previous models typically had fewer additional layers, with a maximum of two additional Dense layers.The training settings, such as the number of epochs, batch size, optimizer, loss function, and callbacks were kept constant with respect to the first model since it had the best accuracy and Similar to the previous models, the base layers of the InceptionV3 model remain frozen, preventing their weights from being updated during training to leverage pre-trained features.

The training accuracy of this model is 91.67%, which is lower than the training accuracies of the previous models that achieved 94.81% and 95.04%, respectively. However, he validation accuracy for this model stands at 91.93% and The validation loss of 0.1898 is comparable to the previous models, indicating reasonable generalization capability.

### 6.3.1 Real Time Testing

Even though the accuracy was not improved by introducing several dense layers, the validation loss was slighty reduced. The testing of this model is shown in Figure 10a and 10b. [9].




(a) Test 3 : Open eyes

(b) Test 3 : Closed eyes

Figure 10: Comparison of Test 3

## 6.4 Discussions

The conducted experiments involved training and evaluating models for eye detection or drowsiness prediction, revealing nuanced insights into their performance. While these models showcased relatively high training accuracies, ranging from approximately 91.67% to 95.04%, the observed validation and test accuracies were slightly lower, varying between 83.74% to 92.60%. This discrepancy suggests potential limitations in the models' ability to generalize effectively to unseen or real-world data. Furthermore, the occurrence of early stopping at different epochs across models indicated the possibility that the models reached their optimal performance levels at distinct stages during training, potentially impacting their convergence and final predictive capabilities.

Even when the real time testing produced good results, further critical assessment of the experiment design, it becomes evident that while the fundamental aspects of model construction, data preprocessing, and evaluation were adhered to, certain limitations might have affected the models' performance. In terms of the architecture, the exploration

[9]Note: Proper consent was obtained from the person for the experiment

Table 1: Evaluation metrics

| Experiment No. | Training Accuracy | Test Accuracy | Patience value |
|---|---|---|---|
| 1 | 95.04% | 86.04% | 7 |
| 2 | 94.81% | 83.92% | 14 |
| 3 | 91.67 | 83.74% | 7 |

of a limited set of model architectures might have restricted the discovery of more effective structures for eye detection tasks.

# 7 Conclusions and Future Works

The research aimed to develop and evaluate models for drowsiness prediction, focusing on achieving high accuracy in real-time scenarios. The primary objectives were to construct models capable of accurately distinguishing between open and closed eyes, assess their generalization on unseen data, and evaluate their effectiveness in practical applications

## 7.1 Key findings:

The research partially succeeded in achieving its objectives. While the constructed models showcased promising training accuracies, ranging from 91.67% to 95.04%, their performance on validation and test datasets was slightly lower, ranging from 83.74% to 92.60%. This suggests limitations in generalization, impacting their reliability in real-world applications.

Early stopping occurred at different epochs across models, indicating varied convergence and optimal performance stages during training.

Comprehensive hyperparameter tuning, exploring diverse model architectures, and leveraging advanced regularization techniques to enhance model generalization. Exploring transfer learning approaches or more sophisticated neural network architectures for improved feature extraction is another dimension that can produce potential results in this field.

The potential for commercialization lies in deploying robust eye detection or drowsiness prediction systems in real-time applications, such as driver assistance systems or human-computer interaction devices. Improving the model's accuracy and reliability through further research could enhance their practical usability in safety-critical scenarios

# References

Ali, A., Yaseen, M., Aljanabi, M., Abed, S. A. and Gpt, C. (2023). Transfer learning: A new promising techniques, *Mesopotamian Journal of Big Data* .

Chellappa, A., Reddy, M. S., Ezhilarasie, R., Suguna, S. K. and Umamakeswari, A. (2018). Fatigue detection using raspberry pi 3, *International Journal of Engineering & Technology* **7**(2.24): 29–32. Website: `www.sciencepubco.com/index.php/IJET`.

Chirra, V. R. R., Uyyala, S. R. and Kolli, V. K. K. (2019). Deep cnn: A machine learning approach for driver drowsiness detection based on eye state, *Rev. d'Intelligence Artif.* .

Dua, M., Shakshi, Singla, R., Raj, S. and Jangra, A. (2021). Deep cnn models-based ensemble approach to driver drowsiness detection, *Neural Computing & Applications* **33**(8): 3155–3168.
**URL:** *https://doi.org/10.1007/s00521-020-05209-7*

Fouzia, Roopalakshmi, R., Rathod, J., Shetty, A. and Supriya, K. (2018). Driver drowsiness detection system based on visual features, *Proceedings of the International Conference on Inventive Communication and Computational Technologies, ICICCT 2018*, Institute of Electrical and Electronics Engineers Inc., Coimbatore, Tamil Nadu, India, pp. 1344–1347.

Hashemi, M., Mirrashid, A. and Beheshti Shirazi, A. (2020). Driver safety development: Real-time driver drowsiness detection system based on convolutional neural network, *SN Computer Science* .

K, R. R., U, N. K., Isak, R. C. T., S, S. K., Sunny, S. and K, V. E. (2017). Drowsiness detection and rescue system, *International Journal of Science and Research (IJSR)* **6**(3). Licensed Under Creative Commons Attribution CC BY.
**URL:** *http://www.ijsr.net*

Kim, P. (2017). *Convolutional Neural Network*, Radiopaedia.org.

Kulkarni, S. S., Harale, A. D. and Thakur, A. V. (2017). Image processing for driver's safety and vehicle control using raspberry pi and webcam, *IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)*, IEEE, Chennai, pp. 1288–1291.

Maior, C., Moura, M., Santana, J. M. M., do Nascimento, L. M., Macedo, J., Lins, I. and Droguett, E. (2018). Real-time svm classification for drowsiness detection using eye aspect.

R, J. and Jacob, C. (2022). Deep cnn based approach for driver drowsiness detection, *2022 IEEE International Power and Renewable Energy Conference (IPRECON)*, IEEE.

S., Ramli, R., Azri, M. A., Aliff, M. and Mohammad, Z. (2022). Raspberry pi based driver drowsiness detection system using convolutional neural network (cnn), *2022 IEEE 18th International Colloquium on Signal Processing & Applications (CSPA)*, IEEE.

Singh, P., Upadhyay, M., Gupta, A. and Lamba, P. S. (2021). *CNN-Based Driver Drowsiness Detection System*, Springer.