Transportation:

Enhancing Road Safety through a Driver Fatigue Detection System

50.035 Computer Vision

(Track A. Design Track)

Check Off 3

Stefanie Tan Hui Zhen 1006073

Mubaraquali Muhammed Sufyanali 1006394

Sean Phay Wei Xiang 1005969

Ong Zheng Han 1005867

1 Introduction

1.1 Background Information

Road traffic accidents are a growing global concern, with human error—such as driver fatigue and distracted driving—contributing significantly to the problem [1]. Driver fatigue is particularly dangerous, as it impairs reaction times, decision-making abilities, and alertness, increasing the likelihood of accidents [2].

In Singapore, road safety is becoming increasing challenging due to the rising number of vehicles. The Singapore Police Force reported a troubling increase in traffic fatalities, from 104 in 2022 to 131 in 2023 [3]. While driver fatigue is not the primary cause of road accidents, it remains a significant contributing factor, particularly when compounded by other risky behaviours such as driving under the influence.

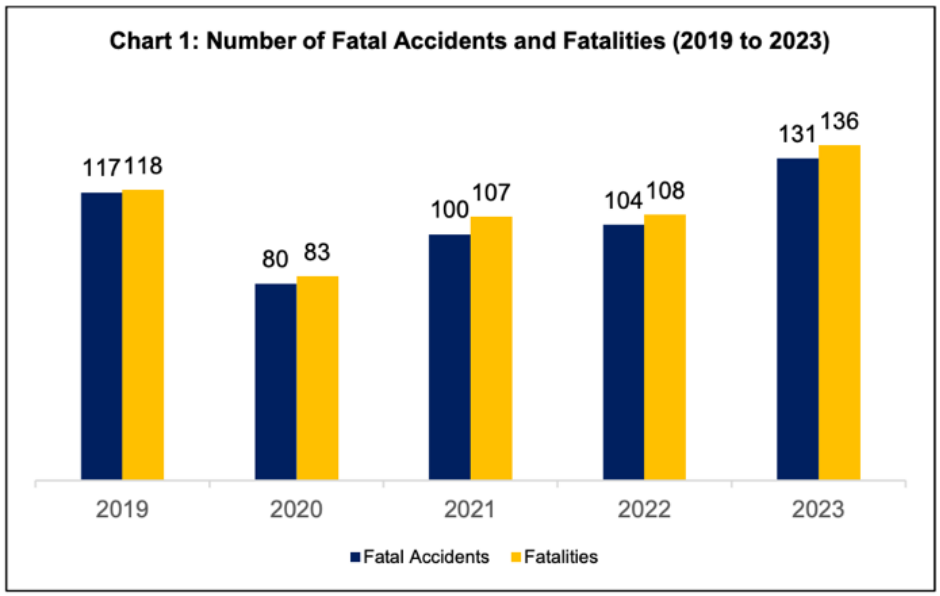

Figure 1: Number of Fatal Accidents and Fatalities (2019 to 2023) [3]

Research has also revealed the prevalence of fatigue among drivers in Singapore. For instance, a study found that 32.9% of taxi drivers reported experiencing sleepiness and fatigue while on the job [2]. These findings highlight the critical need for initiatives that help drivers monitor and address their fatigue levels, thereby reducing the risk of accidents.

1.2 Proposed Solution

The proposed solution in this paper offers a novel approach by focusing on the early detection and intervention of driver fatigue, aiming to prevent accidents and potentially save numerous lives.

1.3 Problem Statement

Drivers are often unaware of their fatigue levels, continuing to drive in a compromised state, which increases the risk of accidents. This project aims to develop a driver fatigue detection system that alerts drivers to their fatigue levels in real-time, providing an opportunity to prevent fatigue-related incidents and enhance road safety.

2 Solution Overview

Our proposed solution focuses on developing a Driver Fatigue Detection System using computer vision concepts aimed at enhancing road safety by addressing drowsy or inattentive driving. The system applies techniques from image processing and facial recognition to use real-time camera-based monitoring to analyze the driver's facial and physical cues, such as eye closure and yawning. These cues are analyzed using algorithms that detect patterns indicative of fatigue or distraction.

When the system detects potential drowsiness, it triggers an alarm to alert the driver to act in real-time, prompting them to take breaks or corrective actions. Implementing such alarm systems will lead to safer roads by reducing the likelihood of accidents caused by human factors like fatigue. This not only protects drivers but also passengers and pedestrians, contributing to overall road safety and societal well-being.

3 Methodology

The system is designed to leverages computer vision and machine learning techniques for real-time detection of drowsiness, ensuring road safety by alerting drivers when signs of fatigue are detected.

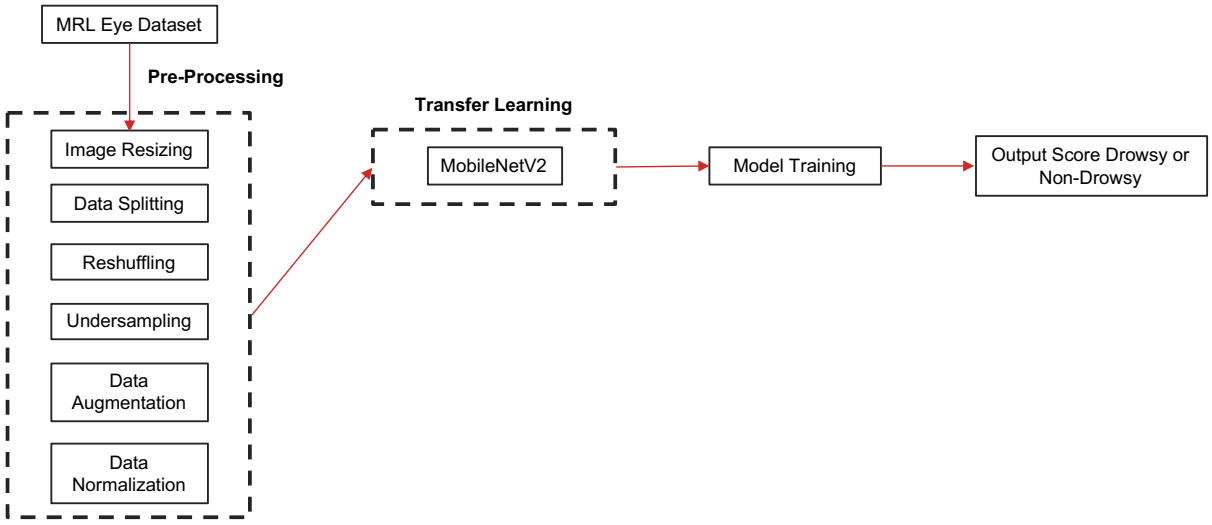

Figure 2: CNN Model Architecture

The system leverages the MRL Eye Dataset to train the model effectively. The dataset undergoes a comprehensive preprocessing pipeline that includes:

1. *Image Resizing*: Standardizing image dimensions for consistency.
2. *Data Splitting*: Dividing the dataset into training, validation, and testing sets.
3. *Reshuffling*: Randomizing data order to prevent bias.
4. *Undersampling*: Balancing class distributions to address data imbalance.
5. *Data Augmentation*: Enhancing variability by applying transformations like rotation and flipping.
6. *Data Normalization*: Scaling pixel values for compatibility with the model.

The pre-processed data is then used for transfer learning with MobileNetV2 architecture, a lightweight and efficient convolutional neural network (CNN). The trained model produces a classification score, identifying whether the individual is **drowsy** or **non-drowsy** with high accuracy.

3.1 Training Dataset (MRL Eye Dataset)

The dataset used for this project is taken from Kaggle's MRL Eye Dataset [4] which consist of eye images captured in grayscale format (8-bit) and labelled to indicate whether the subject is in a drowsy or non-drowsy state. The images are organized into subdirectories within a main folder as "Open-Eyes" and "Close-Eyes". Labels are determined based on the 16th character of the filenames, with a value of '1' indicating a drowsy state and value of '0' indicating a non-drowsy state. The dataset contains up to 84,000 images with varying conditions, including differences in lighting, eye position, and individual characteristics such as eye shape and size.

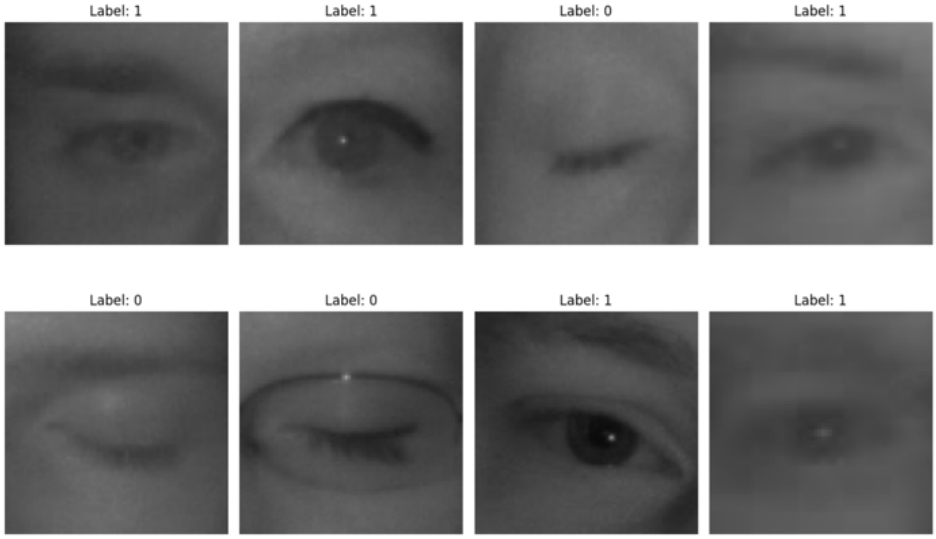

Figure 3 Sample of original images from Kaggle with their respective labels [4]

3.2 Pre-processed Data for Training

All images are resized to 224x224 pixels during preprocessing to ensure compatibility with the model. The dataset is split into three subsets: 70% for training, 15% for validation, and 15% for testing, ensuring effective model training and evaluation. Additionally, data augmentation is applied to enhance the diversity of training images and improve model generalization by including variations such as rotations, flips, shifts, zooms, and shearing. This dataset serves as the foundation for training and evaluating the prototype model to effectively detect drowsiness.

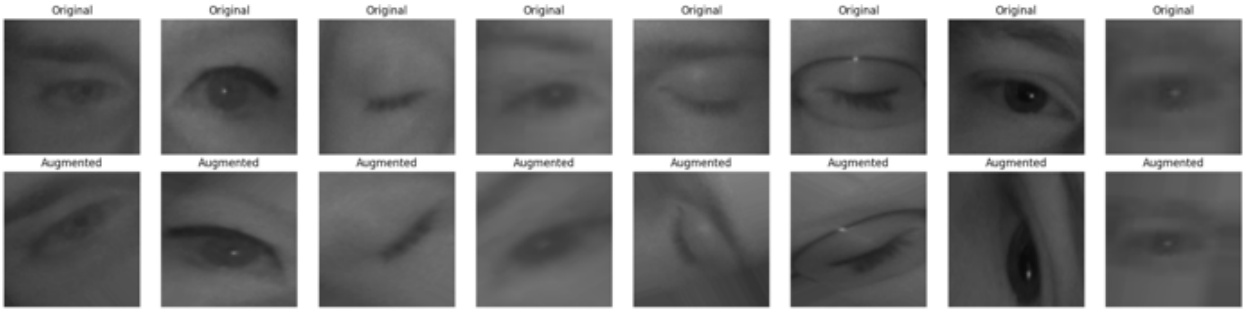

Figure 4: Sample of original images compared to augmented images

3.3 Transfer Learning using MobileNetV2 Model

To develop an efficient and accurate model for classifying drowsiness through eye states (drowsy or non-drowsy), we employed MobileNetV2 as the base model through transfer learning. MobileNetV2 is a lightweight convolutional neural network (CNN) optimized for mobile and embedded vision applications, making it well-suited for tasks requiring high efficiency without compromising performance [5].

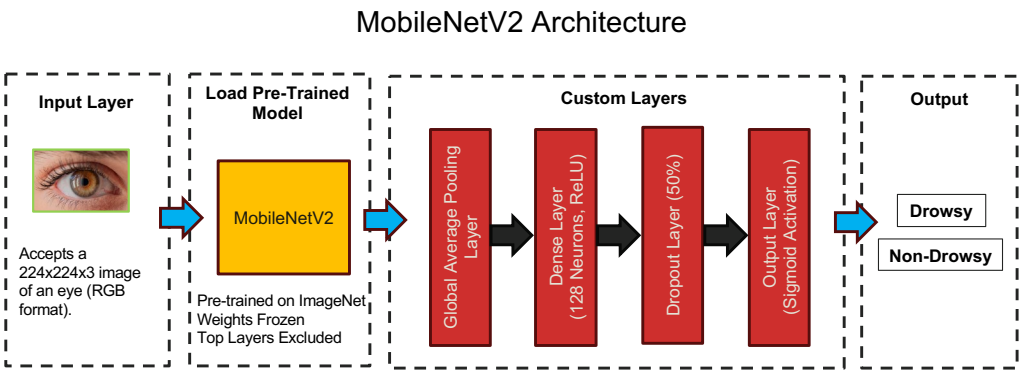

Figure 5: MobileNetV2 Architecture

Base Model Setup

The pre-trained MobileNetV2 model was originally trained on ImageNet dataset, loaded with its pre-trained weights (*weights='imagenet'*), and employed for transfer learning. The top classification layers of MobileNetV2 were excluded (*include_top=False*), allowing us to adapt the model for our specific binary classification task: distinguishing between drowsy and non-drowsy states. The input shape was defined as (*224, 224, 3*) to match the expected input dimensions. To preserve the learned features of MobileNetV2, the base model was frozen (*base_model.trainable = False*).

Custom Classification Layers

After the base model's output, we added a series of custom layers to tailor the model to our requirements:

- *Global Average Pooling Layer*: Condenses the spatial dimensions of the feature maps into a single vector per feature map, reducing the number of dimensionality and improving generalization.
- *Dense Layer (128 Units):* A fully connected layer with 128 neurons and ReLU activation function, introducing non-linearity to the model while learning complex patterns.
- *Dropout Layer (0.5):* Applies a dropout rate of 50% to reduce overfitting by randomly deactivating neurons during training.
- *Output Layer*: A final Dense layer with a single neuron and a sigmoid activation function to output single probability value for binary classification indicating the drowsiness state (drowsy or non-drowsy).

Compilation and Optimization

The model is compiled using the Adam optimizer, which combines adaptive learning rates and momentum for efficient training. The loss function was set to binary cross-entropy, suitable for binary classification tasks, and accuracy is used as the evaluation metric to monitor performance.

This architecture leverages MobileNetV2's pre-trained features while allowing the custom layers to adapt to the specific data and task, achieving a balance between computational efficiency and task-specific accuracy.

4 Prototype System Design

The system design for our prototype integrates multiple stages, ensuring efficient and accurate detection of drowsiness. The architecture is structured around core components: real-time video data, facial landmark detection, fatigue detection metrics, fatigue scoring, and alert system activation.

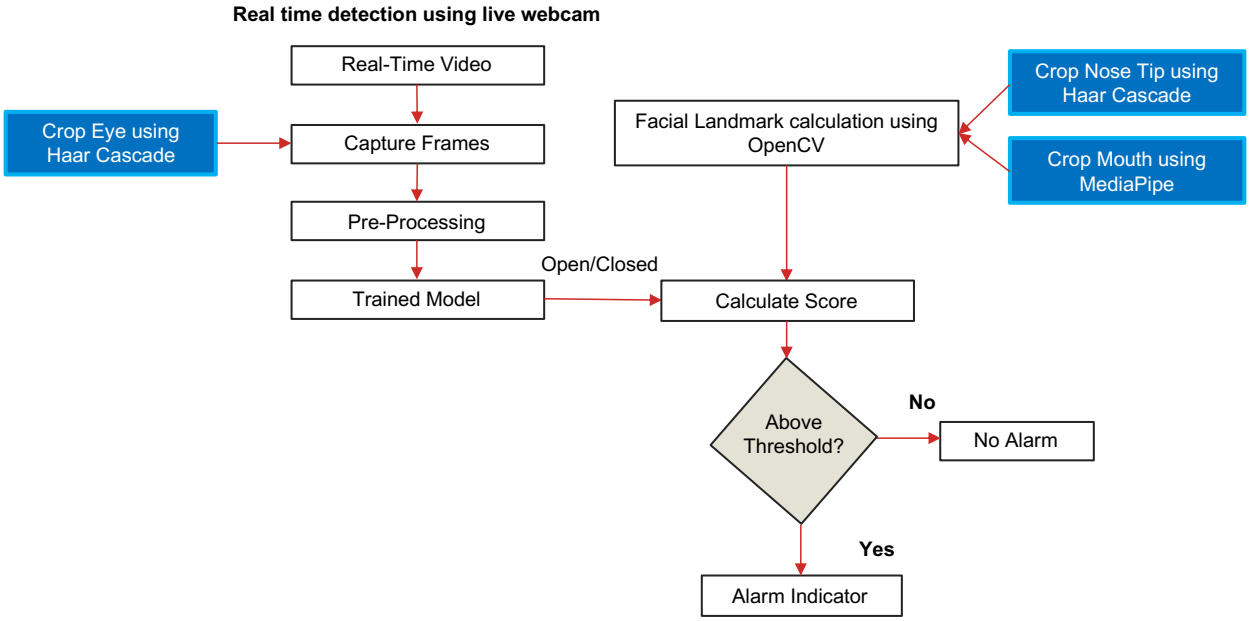

Figure 6: Prototype (OpenCV) Architecture

The prototype architecture diagram illustrates the operational workflow of our real-time fatigue detection system. The webcam captures a real-time video stream, which is processed frame by frame into pre-processed data. Facial landmarks are detected for each frame – Haar cascade classifiers identify the eye and nose regions, while MediaPipe extracts mouth landmarks. These features are then analysed to compute a composite fatigue score based on predefined metrics. If the fatigue score surpasses the predetermined threshold, the system activates an alarm to alert the driver. Otherwise, monitoring continues seamlessly to ensure uninterrupted observation and safety.

4.1 Real-Time Video Data & Capturing Frames

The system utilizes OpenCV for live video feed captured through a webcam as the primary source of input. The data is processed frame-by-frame in real-time, enabling dynamic monitoring of driver behaviour. The video feed provides the raw data necessary for subsequent stages of processing, focusing on extracting facial landmarks.

4.2 Facial Landmarks Detection

The system uses Haar cascades to detect and isolate the driver's face in each video frame and identifies bounding boxes around detected faces. Haar cascade classifiers, pre-trained for facial feature detection, identify key features such as the face, eyes, and nose. Additionally, Mediapipe is utilized to accurately isolate the driver's mouth, enhancing detection efficiency. This process is crucial for defining the region of interest (ROI) within the frames, ensuring that subsequent analysis focuses exclusively on the driver's facial features for accurate fatigue detection.

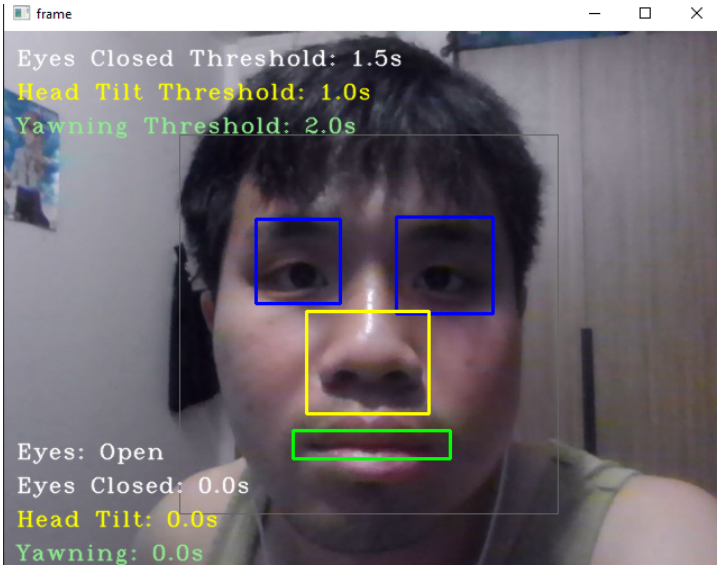

Figure 7: Facial Landmark Detection (Eye Region, Mouth, Nose)

4.2.1 Eye Region Detection

Following face detection, the system uses additional Haar cascade classifiers to identify the left and right eye regions. The classifiers provide bounding boxes around the detected eyes, allowing the system to extract and preprocess these areas for further evaluation. These localized eye regions are resized to 224×224 pixels, converted to RGB format, and normalized pixel values to the range [0, 1] by dividing by 255 for compatibility with the pre-trained MobileNetV2 model. This model predicts whether the eyes are "**open**" or "**closed**," providing critical input for fatigue scoring.

4.2.2 Mouth Detection

The system relies on the Mediapipe FaceMesh model to detect mouth landmarks. Mediapipe is an open-source framework developed by Google for building real-time computer vision pipelines. In this case, the FaceMesh model is used to detect 2D pixel coordinates of the mouth landmarks are extracted for every frame based on their relative positions in the video feed [6]. A predefined list of 15 specific points (landmarks) from the FaceMesh is used to calculate the Mouth Aspect Ratio (MAR) during a yawn. The bounding box around the mouth provides real-time visual feedback.

4.2.3 Nose Detection

Haar cascades are also used to locate the nose region within the face ROI and identifies the position of the nose region with a bounding box around it. The y-coordinate of the nose tip is calculated to monitor head tilt. Changes in the nose tip's vertical position across frames are used as an indicator of head tilting behaviour.

4.3 Fatigue Detection Metrics – Yawn Detection

Yawn detection uses facial landmarks to track the mouth's dimensions in real-time (refer to 4.2.2 Mouth Detection). The Mouth Aspect Ratio (MAR) quantifies the degree of mouth opening, a key indicator of yawning. This ratio is calculated for each video frame and compared against a threshold to detect yawning behaviour. Prolonged yawning increases the yawn score, contributing to the fatigue detection metric.

MAR Formula

The MAR measures the vertical opening of the mouth relative to its horizontal width:

$$\text{MAR} = \frac{(verticle1 + verticle2)}{2 \times horizontal}$$

where $verticle1$ is the distance between inner upper and lower lips, $verticle2$ is the distance between outer upper and lower lips, and $horizontal$ is the distance between the corners of the lips.

A higher MAR typically corresponds to a wider mouth opening, which is characteristic of yawning. The system calculates the MAR for each frame and compares it to a set threshold to detect a yawn. If the MAR exceeds the threshold, a yawn is detected.

Yawning Scoring

If yawning is detected, the yawn score increments by a predefined value. When no yawning is detected, the score decreases gradually, ensuring it doesn't drop below zero.

$$yawn\_score = \max(0, yawn\_score \pm decrement\_value)$$

The yawn score calculated in terms of frames, is converted into seconds for real-world relevance:

$$\text{Yawn Time } (s) = \frac{yawn\_score}{frame\_rate(FPS)}$$

This metric is displayed in real-time on the video feed for user feedback as *Yawning: Yawn Time (s)*.

Yawn Detection

If the yawn time exceeds a predefined time threshold, it signals potential fatigue and it triggers the alarm.

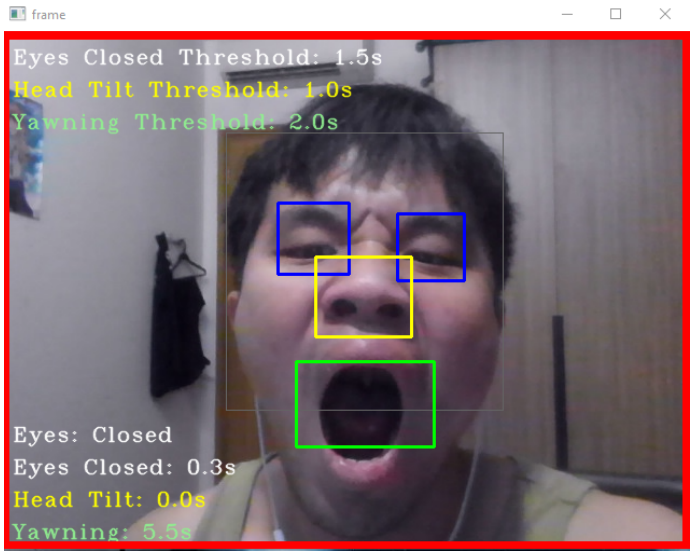

Figure 8: Yawn Detection ("Yawning")

4.4 Fatigue Detection Metrics – Eye State Classification

The pre-trained MobileNetV2 model, fine-tuned on the MRL Eye Dataset, classifies each eye as "**open**" or "**closed**." Using the regions of interest (ROI) for the left and right eyes, determined by the Eye Region Detection process (refer to 4.2.1), the system calculates the eye closure score. This score increases when both eyes remain closed for consecutive frames and decreases otherwise. For clarity, the score is translated into seconds based on the frame rate. These predictions serve as the foundation for the eye closure score, a key indicator of drowsiness.

Model Prediction

Each ROI is fed into the MobileNetV2 model. The output is a single probability (p) for the "open" class:

- $p > 0.5$: Eye is classified as "**open**"
- $p \leq 0.5$: Eye is classified as "**closed**"

Eye Closure Scoring

Condition: Both eyes must be classified as "closed" for consecutive frames to increment the eye closure score.

The eye closed score increases by a predefined increment value for each frame where both eyes remain "**closed**". If either or both eyes are classified as "**open**", the eye closure score is decremented by the same increment value but is ensured to remain non-negative.

$$eye\_closed\_score = \max(0, eye\_closed\_score \pm increment\_value)$$

The eye closure score resets or decreases in order to prevent short-term blinks from accumulating a high score.

The eye closure score, calculated in terms of frames, is converted into seconds for real-world relevance:

$$\text{Eye Closure Time } (s) = \frac{eye\_closed\_score}{frame\_rate(FPS)}$$

This metric is displayed in real-time on the video feed for user feedback as *Eyes Closed: Eye Closure Time (s)*.

Eye Closure Detection

If the eye closure time exceeds the threshold set by the system, the system interprets it as a sign of drowsiness and triggers an alarm.

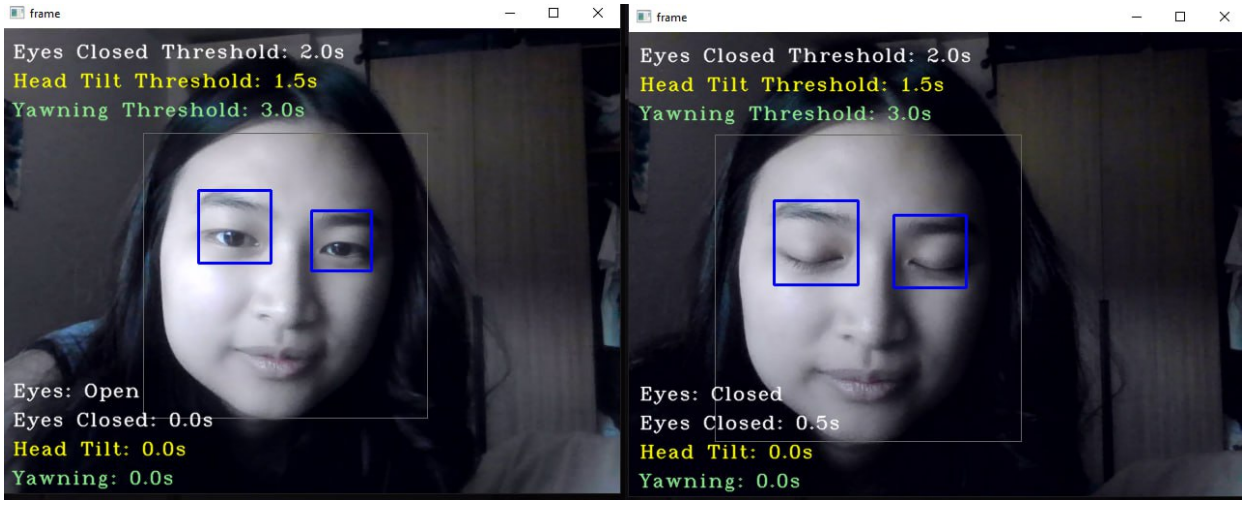

Figure 9: Eye State Classification ("Open" and "Closed")

4.5 Fatigue Detection Metrics – Head Tilting

Head tilting is detected by tracking the nose tip's y-coordinate (refer to 4.2.3 Nose Detection). Changes in the nose tip's y-coordinate across consecutive frames indicate head tilting, which is common during drowsiness or inattention. A tilt score is dynamically updated based on the magnitude of these movements, contributing to the overall fatigue score.

Nose Tip Calculation

This formula identifies the midpoint of the nose bounding box along the vertical axis, which corresponds to the nose tip's approximate position relative to the video frame. The vertical position of the nose tip is calculated as:

$$nose\_tip\_y = y + ny + \frac{nh}{2}$$

where y is the y-coordinate of the face, ny is the y-coordinate of the nose bounding box relative to the ROI, and nh is the height of the nose bounding box.

Head Tilt Scoring

The system tracks the nose tip's y-coordinate across consecutive frames. Significant vertical displacement implies head tilting. A threshold, is used to determine significant movement:

- $|current\_nose\_tip\_y - previous\_nose\_tip\_y| > y\_diff\_threshold$: Head tilt detected
- Otherwise: No head tilt detected

Each frame where a head tilt is detected, it increases the head tilt score by a predefined increment value. If no significant displacement is detected, the score decreases by the same increment but is clamped to a minimum of zero.

$$head\_tilt\_score = \max(0, head\_tilt\_score \pm increment\_value)$$

The head tilt score (in frames) is converted to seconds using the frame rate:

$$\text{Head Tilt Duration } (s) = \frac{head\_tilt\_score}{frame\_rate(FPS)}$$

This metric is displayed in real-time on the video feed for user feedback as *Head Tilt: Head Tilt Duration (s)*.

Head Tilt Detection

If the head tilt duration exceeds a threshold, the system interprets it as a sign of drowsiness and triggers an alarm.

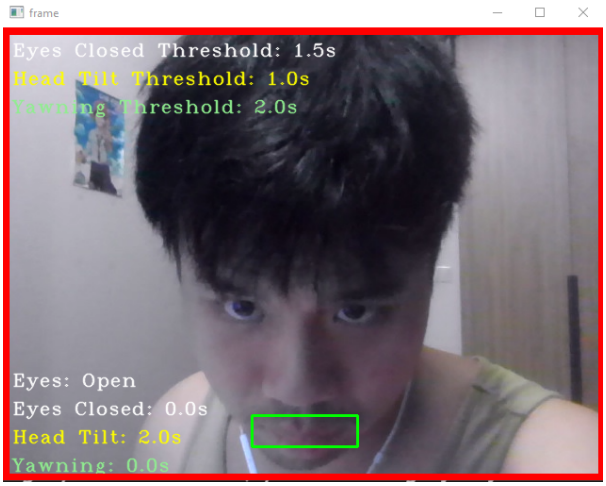

Figure 10: Head Tilt Detection ("Head Tilted")

4.6 Alert System Activation

The Alert System Activation module monitors fatigue detection metrics (eye closure, head tilt, and yawning) in real-time and triggers an alert (an audible alarm by using *pygame.mixer* for playing alarm sounds) when thresholds for any of these metrics are exceeded. This mechanism ensures immediate feedback to counteract drowsiness.

Input Alarm Activation

The alert system continuously monitors the following fatigue detection scores:

- *Eye closure score* (refer to 4.4 Fatigue Detection Metrics – Eye State Classification): Increases when both eyes are classified as "closed" for consecutive frames.
- *Head tilt score* (refer to 4.5 Fatigue Detection Metrics – Head Tilting): Increases when significant vertical displacement of the nose tip is detected across frames.
- *Yawning score* (refer to 4.3 Fatigue Detection Metrics – Yawn Detection): Increases when the Mouth Aspect Ratio (MAR) exceeds the yawning threshold.

Each score is maintained in terms of frames and converted to seconds for interpretability.

Thresholds for activation

For each metric, a predefined threshold in seconds determines whether the behaviour signifies fatigue:

- Eye closure threshold
- Head tilt threshold
- Yawn threshold

These thresholds are dynamically converted into frames based on the video feed's frame rate:

$$\text{Threshold Frame} = \text{Threshold Seconds} \times \text{Frame Rate (FPS)}$$

Alarm Trigger Conditions

The alarm is activated if any one of the following conditions is met:

- Eye closure score exceeds its threshold in frames: $eye\_closed\_score \geq eye\_closed\_threshold\_frames$
- Head tilt score exceeds its threshold in frame: $head\_tilt\_score \geq head\_tilt\_threshold\_frames$
- Yawn score exceeds its threshold in frames: $yawn\_score \geq yawn\_threshold\_frames$

Visual and Audio Alerts

Once the thresholds are breached, the system activates both audio and visual alerts:

- **Audio Alert**: A preloaded alarm sound (*alarm.wav*) is played using the *pygame* mixer. This ensures immediate user attention.
- **Visual Alert**: The video feed displays a red rectangular border around the frame to reinforce the alert.

Reset Mechanism

The alarm continues as long as the scores remain above the thresholds. If scores drop below thresholds (indicating recovery or user alertness), the alert system deactivates automatically.

4.7 Overall Integration

The fatigue detection prototype integrates multiple components, including eye state classification, yawning detection, head tilt detection, and an alert system, to provide real-time drowsiness monitoring. Real-time video data, captured through a webcam are processed to extract facial landmark features such as eyes, mouth, and nose using Haar Cascades and Mediapipe, which are then analysed for fatigue indicators, including eye closure, yawning, and head tilt. Metrics like eye closure score, yawn score, and head tilt score are calculated and compared against predefined thresholds. If any threshold is exceeded, the system triggers an audio alarm and displays a pulsating red border around the video feed to alert the user. This prototype design ensures immediate detection and response to signs of fatigue, providing a robust solution for monitoring driver alertness and enhancing safety.

5 Experimental Details

The fatigue detection prototype was developed to monitor and assess real-time drowsiness based on eye closure, yawning, and head tilting metrics. This experiment aimed to evaluate the prototype's accuracy, responsiveness, and effectiveness in identifying fatigue-related behaviours under controlled conditions.

Accuracy is the ratio of correctly classified instances to the total number of instances.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of images}} \times 100\%$$

5.1 Eye State Classification Evaluation

To train the MobileNetV2-based model, we used the following approach to ensure optimal performance while mitigating overfitting.

Training Process

The model was trained on the dataset using the fit method with the following parameters:

- *Epochs*: A maximum of 10 epochs was set as the upper limit for training.
- *Steps per Epoch*: The number of steps was based on the size of the training data (*len(train_generator)*).
- *Validation Steps*: Similarly, the number of steps for validation was determined by the size of the validation data (*len(val_generator)*).
- *Callbacks*: Early stopping was included as a callback to dynamically manage training duration.

Results

- *Epoch 1*: The model achieved an accuracy of **87.24%** on the training data and **91.52%** on validation data, with a validation loss of **0.2021**.
- *Epochs 2–7*: Training continued, but performance improvements plateaued. Early stopping was triggered at epoch 7.
- **Final Evaluation**: On the test set, the model achieved an accuracy of **91.24%**, demonstrating strong generalization.

This training approach allowed us to balance performance optimization and computational efficiency, minimizing overfitting while achieving a robust accuracy score on unseen data.

5.2 Head Tilt Detection Evaluation

The evaluation was conducted using labelled image datasets and a classification algorithm based on the detection of facial features, particularly the nose tip, to determine head tilt status.

Dataset Preparation

Two datasets were used in the evaluation:

- *Tilt Dataset*: Images of individuals with head tilts.
- *No Tilt Dataset*: Images of individuals without head tilts.

These datasets were dynamically accessed via directory paths to enable compatibility across different testing environments.

Detection Process

- Haar cascades were employed to detect faces and nose tips in each image.
- If the nose tip was not detected, the image was classified as "**Tilt**" by default.
- For detected nose tips, their y-coordinate displacement was analysed.
    - *Tilt*: y-coordinate displacement ≥ 50 pixels.
    - *No Tilt*: y-coordinate displacement < 50 pixels.

Evaluation Metric

Accuracy was computed as the ratio of correctly classified images to the total number of images.

Results

The head tilt detection system achieved an accuracy of **67.38%**, correctly identifying head tilt status for 111 out of 165 images, demonstrating moderate effectiveness.

5.3 Yawning Detection Evaluation

The yawn detection module classifies images into two categories: "Yawn" and "No Yawn," based on the Mouth Aspect Ratio (MAR) computed during feature analysis. The evaluation was conducted under controlled conditions using labelled datasets to assess accuracy and performance.

Dataset Preparation

Two datasets were used for the evaluation:

- *Yawn Dataset*: Images of individuals yawning.
- *No Yawn Dataset*: Images of individuals not yawning.

These datasets were accessed dynamically using specified directory paths for seamless integration and evaluation.

Detection and Classification

The *get_yawn* function analysed each image to compute the MAR and determine whether yawning behaviour was present.

Classification outcomes:

- *Yawn*: MAR value ≥ 0.61 threshold, indicating yawning behaviour
- *No Yawn*: MAR value < 0.61 threshold.

Predictions and MAR values were recorded for each image in the evaluation results.

Evaluation Metric

Accuracy was computed as the percentage of correctly classified images relative to the total number of images.

Results

The yawn detection system achieved an accuracy of **63.64%**, correctly classifying 119 out of 187 images, reflecting moderate reliability under controlled conditions.

6 Conclusion

The fatigue detection prototype successfully integrates real-time monitoring of eye closure, yawning, and head tilting to assess drowsiness with high accuracy and responsiveness. The system demonstrates effective detection and timely alert activation, ensuring immediate feedback to counteract fatigue-related risks. While challenges such as head tilt variability and occasional false positives exist, the prototype design and adaptability of the system provide a solid foundation for further enhancements. With improvements in detection models, personalized calibration, and environmental adaptability, this prototype holds significant potential for practical applications in driver safety where drowsiness detection is critical. Overall, the prototype is a reliable and efficient tool for promoting alertness and preventing fatigue-induced incidents.

7 Future Work

To enhance the functionality and robustness of the drowsiness detection system, several future directions can be explored:

1. *Mobile and Embedded Device Deployment*: Optimizing the system for deployment on mobile devices or embedded platforms like Raspberry Pi can enable portability and integration into vehicles. Efforts to reduce computational overhead while maintaining accuracy would make the system more practical for real-world applications.
2. *Real-Time Alert System Enhancements*: The current alarm system is relatively simple, but in the future, we could develop a more sophisticated real-time alert system that adapts to the driver's behaviour. For instance:
    - *Yawning Timeframe*: The system monitors yawning frequency within a specified timeframe and triggers the alarm if the user yawns repeatedly during this period, indicating potential fatigue.
    - *Incorporating adaptive feedback*: The system could give verbal warnings, such as "Please pull over and take a rest," or prompt the driver to take breaks at regular intervals.

These enhancements would not only broaden the scope of the system but also make it more reliable, adaptable, and effective in preventing accidents due to drowsiness or distraction.

8 Workload and Contribution

A summary of the teams work assignment is shown below in Table 1.

Sean Phay	Data gathering, Yawning fatigue detection, Head tilt detection, Experiment testing, Report writing, Presentation
Zheng Han	Experiment Testing, Presentation
Stefanie Tan	Data gathering, Fatigue scoring, Evaluation of the detection systems, Experiment testing, Report writing, Presentation
Sufi	CNN model, Facial landmark detection, Eye state classification, Head tilt detection, Alarm system activation, Experiment testing, Report writing, Presentation

Table 1: Work Assignment

9 Appendix

9.1 GitHub Code

You can refer to our repository on GitHub using this link: https://github.com/sufi4414/Driver-Fatigue-System/tree/main

In order to run our code, a webcam, internal or external camera is necessary for video input.

Steps to run the code:

1. Run the Code: Execute the main script (detection.py) using Python

   python detection.py

2. Grant Webcam Access:
   - When prompted, allow the program to access your webcam.
   - The system will display a real-time video feed and begin monitoring for signs of fatigue.
3. Terminate the Program:
   - To stop the program, press the q key in the video feed window.
   - The program will release the webcam and close all windows.

Refer to README.md in GitHub repository for debugging and more information on the code.

Expected Behaviour

- The program will analyse your face in real-time, calculating metrics for eye closure, yawning, and head tilting.
- If thresholds for any metric are exceeded, an alarm will sound, and a pulsating red border will appear on the video feed to alert you.

9.2 Prototype Demo Video

You can refer to this link to view the video demonstration (in YouTube) of how the prototype is expected to behave: https://youtu.be/FjkNC11gxMc

References

[1] R. Loh, "The Big Read: To tackle rising fatal traffic accidents and worsening road culture, we need to first understand the problem," 13 May 2024. [Online]. Available: https://www.channelnewsasia.com/today/big-read/big-read-rising-traffic-accidents-road-culture-4328841.

[2] S. M. Lim and S. E. Chia, "The prevalence of fatigue and associated health and safety risk factors among taxi drivers in Singapore," February 2015. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC4350472/.

[3] Singapore Police Force, "Annual Road Traffic Situation 2023," 2023. [Online]. Available: https://www.police.gov.sg/-/media/D4435F72157942D3B323EE4A507D4CFB.ashx.

[4] I. E. Djerarda , "MRL Eye Dataset," 2022. [Online]. Available: https://www.kaggle.com/datasets/imadeddinedjerarda/mrl-eye-dataset.

[5] F. Alzami, M. Naufal, A. A. Harun, W. Sri and A. S. Moch, "Time Distributed MobileNetV2 with Auto-CLAHE for Eye Region Drowsiness Detection in Low Light Conditions," 2024. [Online]. Available: https://ftp.saiconference.com/Downloads/Volume15No11/Paper_46-Time_Distributed_MobileNetV2_with_Auto_CLAHE.pdf.

[6] Google AI, "Face landmark detection guide," Google, 4 November 2024. [Online]. Available: https://ai.google.dev/edge/mediapipe/solutions/vision/face_landmarker.

[7] M. Venkateswarlu and R. R. C. Venkata , "DrowsyDetectNet: Driver Drowsiness Detection Using Lightweight CNN With Limited Training Data," January 2024. [Online]. Available: https://www.researchgate.net/publication/382983988_DrowsyDetectNet_Driver_Drowsiness_Detection_Using_Lightweight_CNN_with_Limited_Training_Data.