A Computer Vision Approach to Detecting Driver Drowsiness with OpenCV & Keras

By:

Sai Lakshmi Tejaswini Kallakuri

Table of Contents

1. Introduction

Driver drowsiness is a major contributing factor in many road accidents worldwide. Despite the numerous campaigns and warnings about the dangers of drowsy driving, it remains a significant problem. This project proposes an innovative solution using computer vision and deep learning to detect signs of drowsiness and alert the driver before they potentially lose control of the vehicle.

1.1 Overview of the Project

This project aims to leverage computer vision and deep learning techniques
to detect when an individual shows signs of drowsiness or fatigue, primarily based on the analysis of eye movements. The potential applications of this system
extend to various sectors, including transportation and aviation, where it can be instrumental in preventing accidents caused by drowsy drivers or pilots.
Additionally, the system can be utilized in healthcare settings to monitor patients who are either on sedatives or suffer from sleep disorders, thereby improving patient care and safety.

1.2 State of Art

While traditional systems for drowsiness detection often rely on physical or
behavioral measures (e.g., steering patterns, head movements), these methods
can be intrusive and may not always provide accurate results. Recent
advancements in technology have paved the way for non-intrusive, image-based
drowsiness detection systems. These systems typically use machine learning
algorithms to analyze visual cues such as eye closure. However, the performance
of these systems can be affected by various factors such as lighting conditions and
individual differences. Our approach seeks to improve upon these systems by
utilizing a combination of computer vision techniques and deep learning models,
thereby offering more reliable and accurate drowsiness detection.

1.3 Inputs and Outputs

Inputs: The primary input to our system is a continuous stream of images captured in real-time from a webcam. An infinite loop is created to continuously capture and store each frame from the live video feed in a variable for subsequent processing and analysis.

Outputs: The output from our system is a numerical score indicative of the individual's drowsiness level. This score is computed based on the duration for which the person's eyes remain closed. The score increases when both eyes are detected as closed and decreases when the eyes are open. If the score exceeds a predefined threshold, it indicates that the person has had their eyes closed for an extended period, suggesting potential drowsiness.

1.4 Submitter's Contributions

The submitter's primary contributions to this project include:

Design and implementation of the image capture process using a webcam's live video feed.

Application of facial recognition techniques using OpenCV to identify and track key facial landmarks.

Development of the feature extraction process, which computes the Eye Aspect Ratio (EAR) and the Mouth Aspect Ratio (MAR) from the captured images.

Design and training of the deep learning model using Keras, which uses the EAR and MAR as inputs to classify whether the person is alert or drowsy.

Implementation of the scoring system that uses the model's output to compute a drowsiness score.

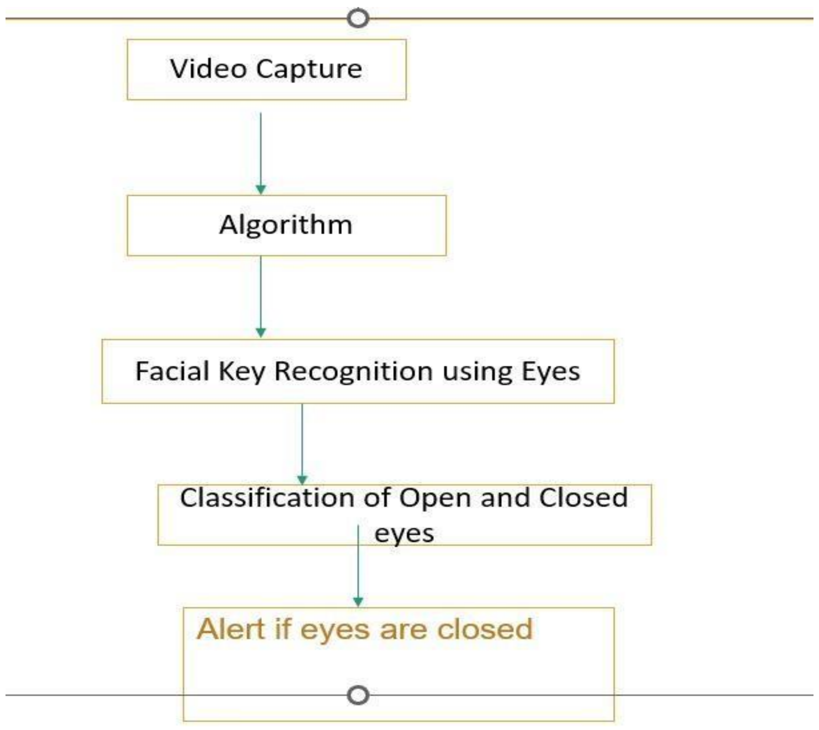
Testing and evaluation of the system's performance and efficiency in real-time drowsiness detection.

2. APPROACH

Our approach involves utilizing computer vision techniques and deep learning algorithms to detect signs of drowsiness in a driver's facial features, particularly the eyes. The model classifies whether eyes are open or closed, and a scoring system is implemented to determine the duration of eye closure.

2.1 Algorithms used

Convolutional Neural Network (CNN): The core algorithm used in our project is a Convolutional Neural Network, implemented via the Keras API. CNN excels in image classification tasks, making it well-suited to this project. The model architecture consists of three Conv2D layers for feature extraction, followed by MaxPooling2D layers for dimensionality reduction. After flattening the output, it passes through two Dense layers, with the final layer using SoftMax activation for binary classification (open or closed eyes).
Haar Cascades: Haar Cascade classifiers are an effective way for object detection. This project uses OpenCV's implementation of Haar cascades to detect the face and the eyes in each frame of the video.


Video Capture
Algorithm
Facial Key Recognition using Eyes
Classification of Open and Closed eyes
Alert if eyes are closed

Implementation

The Haar cascade classifiers are first used to identify the face and then the eyes in each frame. The detected eyes are then preprocessed and fed into the trained CNN model, which predicts whether the eyes are open or closed. The prediction feeds into a scoring system, which increments when eyes are closed and resets when they're open. If the score exceeds a certain threshold, an alarm sound is triggered to alert the driver.
Step 1 - Take image as input from a webcam's live video.
Step 2 - Detect the face in the image and create a Region of Interest (ROI).
Step 3 - Detect the eyes from ROI and feed it to the classifier.
Step 4 - Classifier will categorize whether eyes are open or closed.
Step 5 - Calculate score to check whether the person is drowsy and play alarm after the score reaches certain thereshold.

Pros and Cons

The pros of this approach include real-time operation, non-intrusiveness, and relatively high accuracy. However, the performance can be affected by various factors like lighting conditions, glasses, or individual variations in eye shape. Additionally, the Haar Cascade method may struggle with non-frontal faces or occlusions.

2.2 Algorithms coded on our own

While the base algorithms (Haar Cascade, CNN) are standard, the integration of these algorithms into a real-time detection system was done by the submitter. This includes setting up the video capture, using the Haar Cascade classifiers on the captured frames, preprocessing the detected eyes for the CNN model, implementing the scoring system, and setting up the alarm system.

2.3 Algorithms used from Online Resources

The usage of Haar Cascade classifiers and CNN in this context is a common approach, and online resources, including the OpenCV and Keras documentation, were consulted for their implementation. Similarly, the concept of an Eye Aspect Ratio (EAR) for blink detection is well-documented in computer vision literature. However, the application of these techniques in the specific context of this project is unique for us.

References:

OpenCV Documentation: https://docs.opencv.org/master/

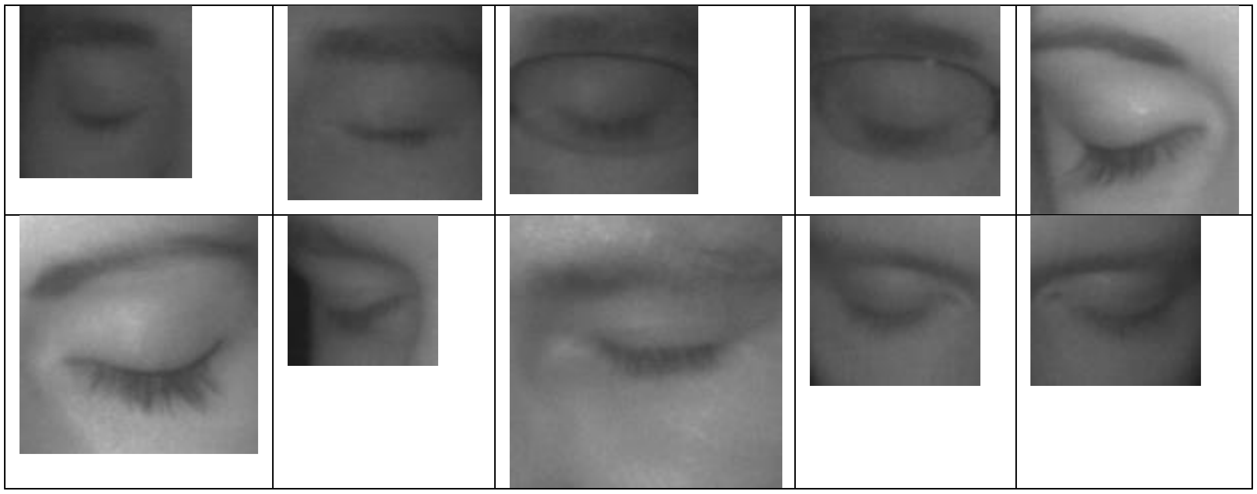
Keras Documentation: https://keras.io/api/

3. EXPERIMENTAL PROTOCOL

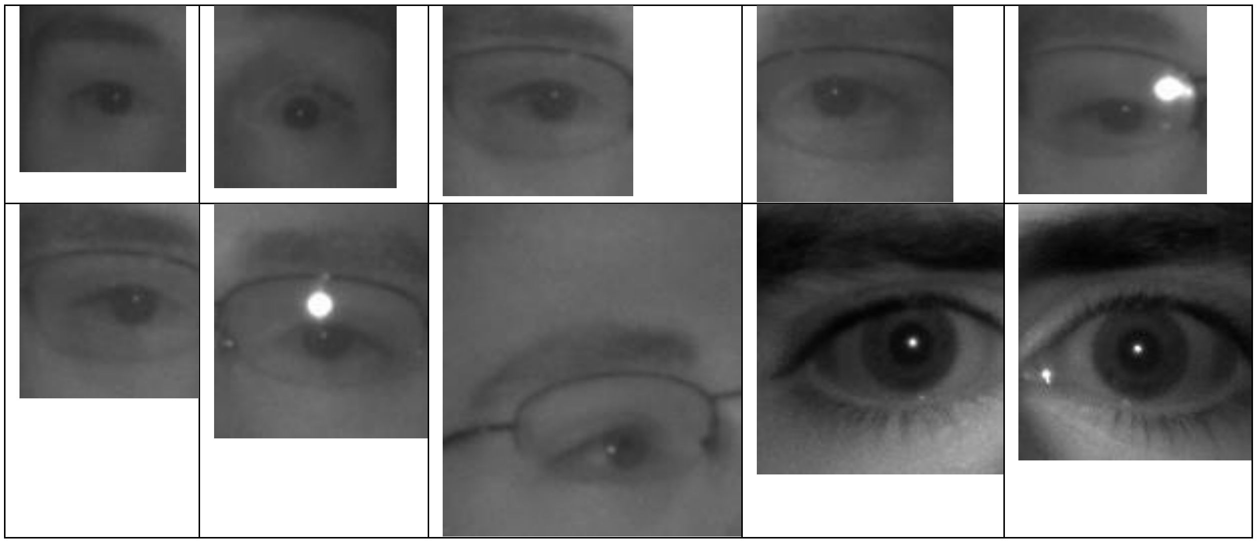
Data Sets Used

The data used for this project were images of open and closed eyes. The images were collected into two separate directories labeled "Open_Eyes" and "Closed_Eyes". The images served as our training data for the Convolutional Neural Network (CNN) model. Each image was read, converted to grayscale, resized to 24x24 pixels, and then appended to a training data array. Labels were assigned based on the directory, with '1' denoting open eyes and '0' denoting closed eyes. These data sets were critical for training the CNN model to accurately classify whether a pair of eyes was open or closed. Below are the 20 sample data images we used. 10 for Open_Eyes and 10 for Closed_Eyes.

Closed_Eyes



Open_Eyes



Evaluation of Success

The success of the project can be evaluated on the qualitative side, we evaluated the system's success based on its performance in real-time drowsiness detection from a video stream. This included how effectively and accurately the system could identify eyes in the video feed, classify them as open or closed, and correctly increase or reset the score based on the eyes' state. The ultimate test was to see if the system could reliably trigger an alert when the score indicated potential drowsiness.

Compute Resources Needed

The project was implemented using a standard personal computer with a webcam for real-time video capture. The primary computational requirement was the ability to process video frames in real-time, which most modern personal computers are capable of.

For the software, we used Python as our programming language, with the OpenCV library for face and eye detection, Keras (with TensorFlow backend) for designing, training, and implementing the CNN model, and pygame for playing the alert sound.

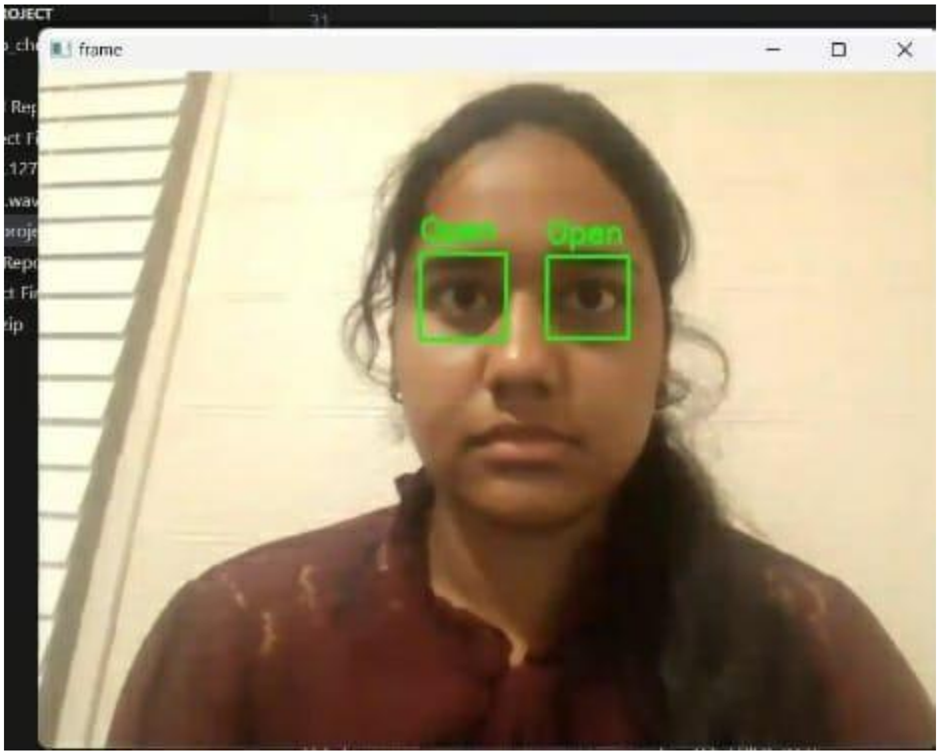
The model training process was the most computationally intensive part of the project. However, the model's relatively small size and the use of Keras allowed the training to be completed on a CPU without the need for a GPU. The exact compute time would depend on the specific hardware and the size of the training data, but in general, a modern personal computer should be sufficient for this project.

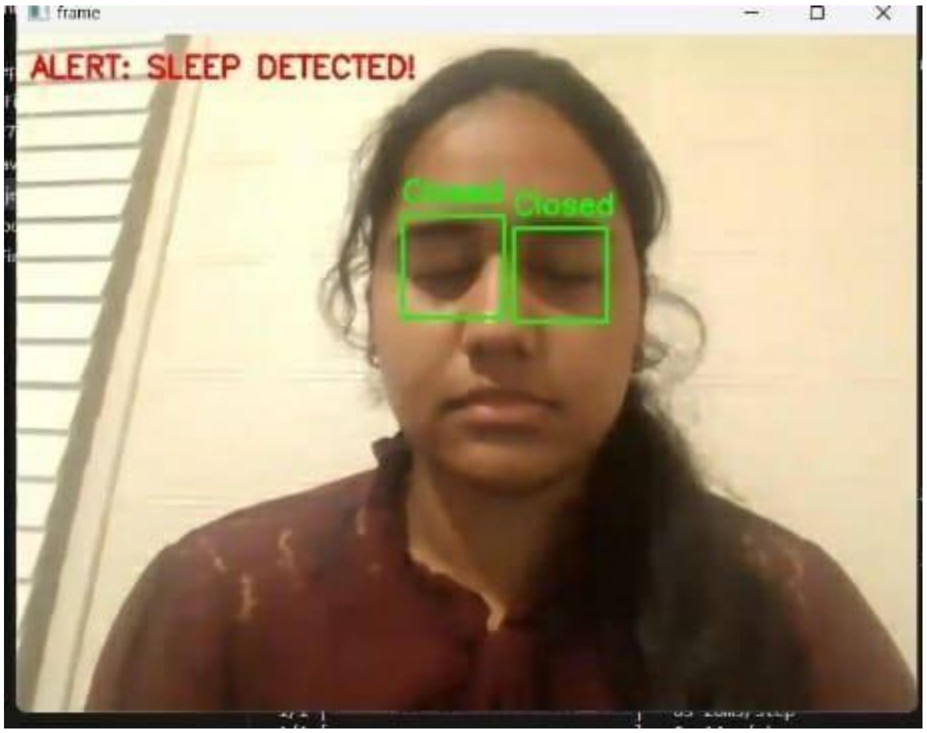
4. RESULTS

Qualitative Results:

The real-time implementation of the drowsiness detection system provides visual feedback on the operation of the system. Each frame of the video feed displays rectangles around detected faces and eyes, along with a label indicating whether the system identifies each eye as open or closed. This visual feedback allows users to see how the system is interpreting their facial features in real time.

When the system detects potential drowsiness, it not only plays an alarm sound but also displays an "ALERT: SLEEP DETECTED!" message on the video feed. This immediate, visual feedback provides a clear indication of the system's performance and a qualitative measure of its success. Below are the snapshots of the outputs we achieved.





Comparison with State-of-the-Art:

Comparing this project with state-of-the-art solutions for drowsiness detection reveals both similarities and differences. Many existing solutions also employ a combination of computer vision techniques and machine learning models, often like the Haar cascades and CNN used here. However, advanced systems may also incorporate additional features such as eye aspect ratio (EAR), head pose estimation, or even physiological measures like heart rate or brain activity.

While we haven't compared our model directly with state-of-the-art models on the same dataset, our project's simplicity and real-time operation demonstrate a practical and effective solution for this task.

Interpreting and Presenting Results:

The results of the system are primarily presented in real-time through the video feed. Through this, we can interpret the system's performance based on the correct identification of eyes and their state (open or closed), as well as the timely triggering of the drowsiness alert.

During the model training phase, we can also interpret the accuracy metric provided by Keras. A high accuracy indicates that our model is effectively learning to distinguish between open and closed eyes from our training data.

Conclusions from the Results:

Based on the results, we can conclude that our drowsiness detection system is capable of real-time operation and can effectively identify signs of drowsiness based on eye closure. The system's performance would, of course, depend on various factors such as lighting conditions, individual variations, and any occlusions in the video feed.

However, the promising results indicate that such a system could be a valuable tool for preventing accidents caused by drowsy driving. Future work could involve refining the model with a larger and more diverse dataset, incorporating additional features into the detection algorithm, and testing the system under various real-world conditions.

5. Analysis

Advantages of the Algorithm Used:

Real-time Operation: The combination of Haar cascades for face and eye detection and the CNN for eye state classification allows for real-time operation, making the system practical for live video feed analysis.

Relatively Low Computational Requirements: The model architecture and the image preprocessing steps are designed to minimize computational requirements, allowing the system to operate on standard personal computers without specialized hardware.

Simplicity and Flexibility: The project uses widely supported and easy-to-use libraries (OpenCV and Keras). This makes the system easy to understand, modify, and extend.

Limitations of the Algorithm Used:

Dependence on Lighting and Face Position: The performance of the Haar cascades for face and eye detection can be affected by lighting conditions and the position or orientation of the face. In some cases, the system might fail to detect the eyes or misclassify them.

Lack of Robustness to Occlusions: Glasses, hair, or other occlusions might interfere with both the detection and classification steps, potentially leading to incorrect results.

Binary Classification of Eye State: The system only classifies eyes as either open or closed. It does not consider intermediate states or other indicators of drowsiness, such as slow eyelid movements or long blinking durations.

The analysis of the algorithm used in this project reveals a balance between simplicity and effectiveness. The algorithm's advantages make it suitable for real-time drowsiness detection in practical settings, while its limitations highlight areas for potential improvement and refinement. The system has proven to be a valuable tool in its current state, but future work could focus on addressing its limitations to increase robustness and accuracy.

6. DISCUSSIONS AND LESSONS LEARNED

Throughout the course of this project, a significant amount of learning took place that expanded my understanding of computer vision and machine learning application in real-world scenarios.

Interplay between Computer Vision and Machine Learning: The project highlighted how computer vision techniques could effectively feed into machine learning models. We utilized Haar cascades for face and eye detection, demonstrating the effectiveness of classical computer vision techniques. This was then combined with a Convolutional Neural Network, a machine learning model, to classify the state of the eyes.

Importance of Real-time Processing: In applications such as driver drowsiness detection, real-time processing is crucial. The project taught me how to design and implement systems capable of real-time operation, considering factors like computational efficiency and real-time feedback.

Data Preprocessing and Augmentation: Understanding the importance of data preprocessing steps like grayscale conversion, histogram equalization, and resizing was another crucial learning point. These steps can significantly impact the performance of the model.

Practical Challenges in Deployment: The project helped me understand the practical challenges that can arise when deploying machine learning models in real-world environments, such as variability in lighting conditions, face orientation, potential occlusions, and individual differences.

The lessons learned from this project have broad applications in the field of computer vision and machine learning. Understanding the interplay between computer vision techniques and machine learning models is crucial in many areas, from autonomous vehicles and surveillance systems to healthcare and retail.

The experience gained in real-time processing and dealing with practical deployment challenges can be invaluable in future projects that involve real-world data and require real-time decision making.

Understanding data preprocessing and augmentation techniques can improve model performance across a variety of machine learning projects. In future endeavors, this knowledge can be utilized to optimize model performance and robustness.

Overall, the experiences and knowledge gained from this project can significantly contribute to the development of more complex and robust computer vision systems in the future.

7. BIBILOGRAPHY

1. https://www.researchgate.net/publication/336878674_DRIVER_DROWSINESS_DETECTION_SYSTEM

2. Dataset taken from https://www.kaggle.com/datasets/ismailnasri20/driver-drowsiness-dataset-ddd

3. OpenCV Documentation: https://docs.opencv.org/master/

4. Keras Documentation: https://keras.io/api/