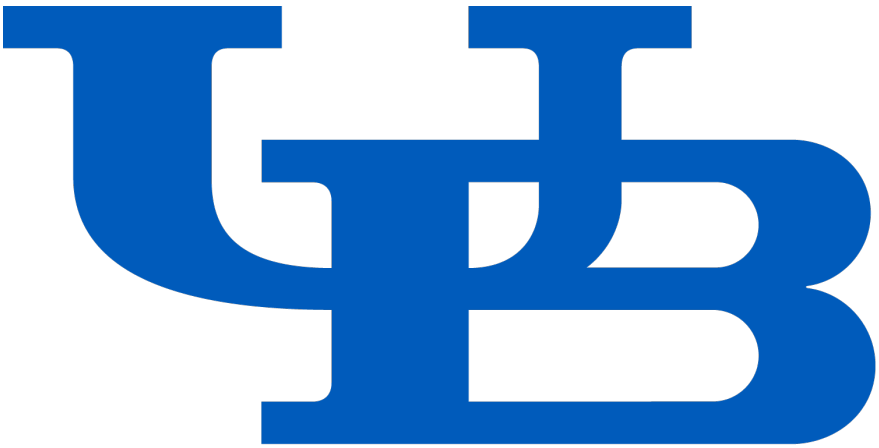
Project Report
CSE573: Computer Vision and Image Processing

Instructor: Dr. Nalini Ratha



Real-Time Drowsiness Detection System

Shailesh Mahto – 50540379

1. Overview

A real-time drowsiness detection system has been developed. The system takes the constant feed of a user (for example, a car driver) as input and alerts the user whenever they show signs of drowsiness. The model uses the blink patterns of the user to identify the user's drowsiness. For this, the system uses the blink pattern classified over a duration of time to identify patterns of drowsiness in the user. The model trained produced an accuracy of 95% on the test set. However, the data used for training was limited to certain demographics as a proof-of-concept. The system was then validated on the publicly available DROZY dataset, where it showed an accuracy of 88%. The model showed promising results but can be improved to include more demographics for generalization or customized for individuals.
The state-of-the-art model produced an accuracy close to 94% on the Drozy dataset. The below table shows the accuracy achieved by the state-of-the-art models using various strategies and datasets.

Table 1: Drowsiness detection approaches and accuracies for state-of-the-art models [1]

Approach	Source	Calculated Parameter	Technology	Dataset	Accuracy	Restriction
[13]	Eyes	Eyes state	Image processing	BioID [28], FEI [29], GI4E [30] and ICPR [31]	BioID: 99.90% FEI: 94.00% GI4E: 99.60% ICPR: 96.00%	Wearing sunglasses
[14]	Eyes	Blinking duration	Viola-Jones face detector	NO	92.85%	Wearing sunglasses
[15]	Eyes	Blinking duration, blinking rate, and head angle	Haar cascade classifier and template matching	NO	97.86%	Wearing sunglasses
[16]	Eyes	Blink patterns	Dlib library and machine learning	DROZY [32]	94.44%	Wearing sunglasses
[17]	Eyes	Eyes state	Haar cascade classifier and machine learning	JZU [33]	94%	Wearing sunglasses
[18]	Eyes	Blinking duration changes	Image processing	NO	-	Wearing sunglasses
[21]	Face	Face landmark points	Dlib library and machine learning	NTHU [34]	81%	-
[2]	Head, eyes, and mouth	Regions of face, eyes, and mouth	machine learning	-	94.8%	-
[19]	Eyes	PERCLOS and blinking duration	Template matching	NO	-	Wearing sunglasses
[22]	Head and eyes	PERCLOS, blinking duration, head tilt, and gaze orientation	Image processing	NO	-	-
[23]	Mouth	Yawning	Image processing	NO	-	Mouth covering, hand on mouth
[24]	Eyes and mouth	Blinking duration and yawning	Viola-Jones and image processing	NO	-	-
[26]	Mouth	Yawning	Viola-Jones and image processing	NO	-	-
[27]	Breathing	Breathing rate	Image processing	NO	-	Clothes

The system developed in this project is inspired by the system in row 4 that used the blink patterns for solving the objective. This model has an extensive feature extraction process which gives more control and interpretability of the model.
Other approaches use yawning, eyes state, head angle or breathing rate for classification. However, most of the models have issues with the existence of sunglasses. Using yawning also produces false positives since talking, and laughing can also resemble yawning in terms of the mouth position.

Since this was a solo project, the entire project was developed by my individual effort. The code for this implementation is also not available in open source, so the concept was entirely implemented by me.

2. Approach

The real-time drowsiness detection system can be broken down into 3 key sections. Firstly, the eye position is detected. Secondly, the Eye-Aspect-Ratio (EAR) is calculated, and the blink type is classified. Thirdly, the blink pattern over a certain time frame is checked, and if certain patterns are observed, then the user is alerted for being drowsy in real-time.

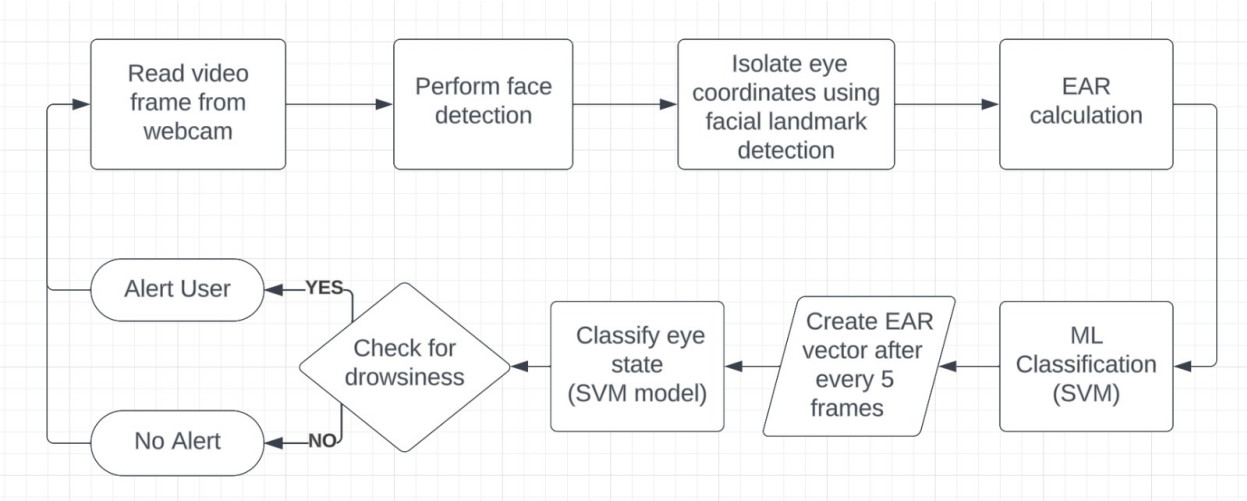


Fig 1: System flow chart for real-time drowsiness detection

2.1. Eye Position Calculation

To initiate the creation of the real-time drowsiness classification model, the initial task is to identify the user's eyes. To achieve this, the dlib library [3] was employed for facial landmark detection, utilizing Histogram of Oriented Gradients (HOG) and linear Support Vector Machine (SVM) [4]. These landmarks exhibit adaptability to diverse human faces and remain unaffected by rapid facial movements.

The code for ear calculation is shown below.

```
detect = dlib.get_frontal_face_detector()
predict =
dlib.shape_predictor("models/shape_predictor_68_face_landmarks.dat")
(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_68_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_68_IDXS["right_eye"]

def eye_aspect_ratio(eye):
```

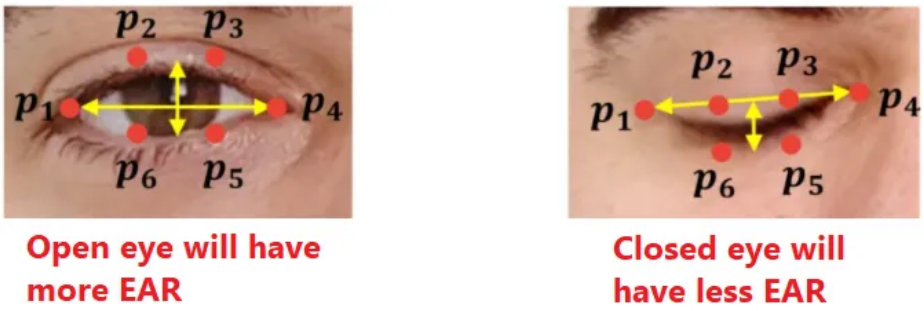
```
      A = distance.euclidean(eye[1], eye[5])
      B = distance.euclidean(eye[2], eye[4])
      C = distance.euclidean(eye[0], eye[3])
      ear = (A + B) / (2.0 * C)
      return ear

def get_ear_from_frame(frame):
    frame = imutils.resize(frame, width=540)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    subjects = detect(gray, 0)
    ear = None
    # find faces in frame
    for subject in subjects:
        shape = predict(gray, subject)
        shape = face_utils.shape_to_np(shape)#converting to NumPy
Array
        leftEye = shape[lStart:lEnd]
        rightEye = shape[rStart:rEnd]
        leftEAR = eye_aspect_ratio(leftEye)
        rightEAR = eye_aspect_ratio(rightEye)
        ear = (leftEAR + rightEAR) / 2.0
    return ear
```

dlib.get_frontal_face_detector - is a function provided by the dlib library in Python for face detection. Specifically, it returns an object that represents a face detector trained to detect frontal faces in images. The dlib library is commonly used for computer vision tasks, including facial landmark detection and face recognition.

The dlib.shape_predictor with the argument "shape_predictor_68_face_landmarks.dat" is used to load a pre-trained facial landmark predictor model in the dlib library. This model is specifically trained to identify 68 facial landmarks on a human face. Facial landmarks are key points on the face, such as the corners of the eyes, nose, mouth, etc. Only the landmark coordinates related to the eyes are stored for further calculation. However, the model showed challenges with users who wear vision-corrective glasses. Since this was a con for the model, the dataset was limited to users that do not wear glasses for implementing this proof-of-concept.

Eye-Aspect-Ratio (EAR) is the ratio between the width and the height of the eyes. This is a way of quantifying the openness of the eyes.



$$EAR = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

Fig 2: Eye landmarks used for EAR calculation

2.2. Eye-Aspect-Ratio Calculation over temporal space

EAR value is calculated for each frame of the video feed. Then EAR from 15 consecutive frames are concatenated to create the user's state feature. The research by Bacivarov et al., 2008 [5] suggested this duration encompasses the blink duration across various user activities. This composite feature, consisting of 15 consecutive EARs, is categorized into three groups: open eyes, short blink, and long blink (coded as 0, 1, and 2, respectively). Following a similar approach to Souto Maior et al. (2018) [6], a blink is acknowledged only if eyelid closure occurs in the five central frames of the state feature (from the 6th to the 10th frame within the 15 selected frames). This methodology prevents the double counting of the same blink or more during real-time model execution, where inputs are selected every five frames.

2.3. Drowsiness Detection

An SVM machine learning model is used to predict the eye state of the user out of three states – open eyes, short blink or long blink. The methodology used for collecting the training data and the model training has been discussed in detail in the Experimental Protocol section of the report. The system produces new sequence every 5 frames (0.21s) which gives the model a temporal dimension. Detecting a single instance of a 'long blink' is insufficient for representing drowsiness; additional information is necessary for making inferences about drowsiness.

Therefore, in the drowsiness detection process, I opted not to consider the classification of each output in isolation but rather focused on the entire pattern they collectively form. The SVM outputs, presented as a sequence of 0, 1, and 2, are utilized to ascertain whether the user is experiencing drowsiness or not. Drawing insights from relevant literature [7], two distinct rules were applied to define drowsiness:

Rule 1: If, within a 60-second interval, the proportion of long blink outputs to the sum of long blink and short blink outputs (total number of blinks) exceeds 25%.

Rule 2: If there are 5 or more consecutive outputs (SVM predictions) indicating long blink. This scenario occurs when the user keeps their eyes closed.

The code for implementing this core logic for drowsiness is as follows:

```
# function that take the df of ear vectors as input and outputs
whether the user is drowsy
def check_for_drowsiness(df):
    continued_long_blink = sum(df["EYE_STATE"].tail(5)) == 10
    if continued_long_blink:
        return 1
    time_now = datetime.now()
    last_minute_start = time_now - timedelta(minutes=1)

    filtered_df = df[df['TIMESTAMP'] > last_minute_start]
    short_blink_count = sum(filtered_df["EYE_STATE"] == 1)
    long_blink_count = sum(filtered_df["EYE_STATE"] == 2)
    total_blink_count = short_blink_count + long_blink_count
    drowsiness_ratio = 0
    if total_blink_count > 1:
        drowsiness_ratio = long_blink_count / (total_blink_count)
    if drowsiness_ratio > 0.25:
        return 2
    return 0
```

Primarily, the first situation occurs when the user gradually blinks multiple times in a brief timeframe. Existing literature [8] suggests that consecutive long blinks are strongly associated with the initial stages of drowsiness. On the other hand, the second rule addresses instances of continuously closed eyes, indicating that the user keeps their eyes closed for an extended duration. If either of these situations is detected, a visual notification is triggered, warning of the potential state of drowsiness (see Fig. 5 and Fig 6). This message serves to alert users about their condition and mitigate potential lapses in attention. The notification is recurrent in its use.

I took inspiration from the following GitHub repository for EAR calculation code.

- https://github.com/akshaybahadur21/Drowsiness_Detection

However, there is no repository (to the best of my knowledge) that implements the rest of the drowsiness detection system. Hence, I coded the rest of the entire code by myself. However, I used [2] to understand the logical flow of the system before implementing it myself.

3. Experimental Protocol

3.1. Dataset collection

No publicly available dataset had the data in the format that I needed for the training the ML model. Specifically, I needed data of people in two states:

1. While they were engaged in their environment. This would simulate the alert state of the user. Open eyes and short blinks would be the two possible states of the user's eyes in this user state.
2. While the user's eyes were closed to simulate the scenario of the user entering the sleep state.

Such a dataset could not be found online and hence I decided to collect my own data.

To generate data required for model development, an internal experiment was conducted. Participants were instructed to view multiple images on the screen over a period of 20 seconds. Additionally, they were requested to keep their eyes closed over a few seconds, simulating the occurrence of long blinks. The objective if the experiment was to observe the EAR behavior when the subjects had their eyes open and when they blinked (while seeing images), as well as to capture their closed-eye pattern related to long blinks.
Fourteen distinct users contributed to the EAR data, acknowledging that it often varies from person to person. This diversity enables the algorithm to learn from varied situations, encompassing inter-individual differences.

Table 2: Demographics information of subjects who participated in data collection process

Subject ID	Gender	Age	Glasses	Beard/Mustache
1	M	26	No	Yes
5	M	25	No	Yes
6	M	23	No	Yes
10	M	23	No	No
11	M	23	No	Yes
12	M	23	No	No
14	M	33	No	No
15	M	23	No	Yes
16	F	24	No	No
17	F	24	No	No
18	M	25	No	No
20	M	30	No	Yes
21	M	27	No	Yes
23	M	25	No	Yes

3.2. Data Labelling

After the video recordings of the users was captured, the video was split into frames. The EAR value for each frame was stored in a list. This list was referred to for creating the EAR vectors that captured the user's state features.

The EAR vector was created once after every 5 frames were passed. Such an EAR vector consisted of the EAR values for the last 15 frames. A blink was acknowledged only if eyelid closure occurred in the five central frames of the state feature (from the 6th to the 10th frame within the 15 selected frames). This methodology prevents the double counting of the same blink or more during real-time model execution, where inputs were selected every five frames.
The code for creating the EAR vectors using the ears list (the EAR from each frame stored in this list) is as follows:

```
ear_v_len = 15
step_size = 5
ear_vec = []
i = 0
# step_size * i + ear_v_len is the index of last ear value, should not
exceed number  # of frames
while step_size * i + ear_v_len <= frames_num:
    start = step_size * i
    end = step_size * i + ear_v_len
    # print(f"ears len: {len(ears)}, i: {i}, start: {start}, end:
{end}")
    ear_vec.append(ears[start:end])
   i = i+1
```

For labeling the data, the central 5 frames were assessed and if eye closure occurred in those frames, such vectors were identified as short blink ("1"). If there was no blink in the central five frames, such vectors were labeled as open eyes ("0"). Labeling for long blinks ("3") was straightforward because the users had their eyes closed throughout the video.

Anticipated eye behaviors encompass the following:

1. Open eyes - the EAR is sizable and exhibits minimal variation throughout the 15 frames.
2. Short blinks - characterized by a swift decrease during eye closure (occurring within the 5 central frames) and an increase during eye-opening. Hence, high variability throughout the 15 frames.
3. Long blinks - the EAR is small and displays limited variation over the course of 15 frames.

3.3. Model Development

The data from 14 subjects was divided into training data and unseen data sets. For this, the data from 3 users was kept for unseen dataset and 11 users for training set. For each user, the number of short blinks was the lowest. Hence, the EAR vectors for open eyes and long blink were undersampled to match the number of vectors of short blink. This was done for each user separately. The code for creating the final sample data for each user is as follows:

```
users_list = ["person1", "person5", "person6", "person10", "person11",
"person12", "person14", "person15", "person16", "person17",
"person18", "person19", "person20", "person21", "person23"]
for user_nm in users_list:
    df = pd.read_excel(f"data/ear_vecs_per_user/{user_nm}/{user_nm}-
labelled.xlsx")

    df0 = df[df.EYE_STATE == 0]
    df1 = df[df.EYE_STATE == 1]
    df2 = df[df.EYE_STATE == 2]

    # undersample all types of rows to have equal distribution
    df_size = min(len(df0), len(df1), len(df2))
    df0 = df0.sample(n=df_size, random_state=42)
    df1 = df1.sample(n=df_size, random_state=42)
    df2 = df2.sample(n=df_size, random_state=42)

    df = pd.concat([df0, df1, df2], ignore_index=True)
    df = df.sample(frac=1, random_state=42)

    df.to_excel("data/ear_vecs_per_user/" + user_nm + "/" + user_nm +
"-sample" + ".xlsx", index=False)
```

This data was then concatenated for the 11 training set subjects and 3 test set subjects separately.

The training data set consisting of data related to 11 subjects had a total of 1254 rows and unseen dataset had 276 rows with an equal distribution of all 3 categories.

The training set was split further into train and test set with an 80-20 split. An SVM model was trained on the train set and validated using test and unseen data set. Since SVM model uses multiple hyperparameters, Grid Search algorithm was used to find the optimal set of hyperparameters. The best model used the following hyperparameters - {'C': 100, 'gamma': 0.1, 'kernel': 'rbf'}. The best model produced an accuracy of 95% on the unseen data set.

The first indication of success was quantitative. Specifically, it was the model's high accuracy score for predicting the user's eye state. This showed that the model had the ability to predict the user's eye state if the EAR temporal vector was given. The second indication of model's success was qualitive. It was the model's ability to alert the user successfully in multiple attempts of simulating drowsiness. The third indication of success was quantitative. For this, the publicly available dataset called DROZY was used to compare the results against the benchmark results of the state-of-the-art models.

3.4. About DROZY Dataset

To assess the accuracy of the drowsiness model, examinations were conducted using an independent and publicly available database known as the "ULg multimodality drowsiness database," or DROZY [9].

Within the DROZY framework, 14 subjects independently underwent three consecutive 10-minute psychomotor vigilance tests (PVT) on two consecutive days, experiencing increased sleep deprivation induced by acute and prolonged waking. Notably, the total sleep deprivation time between the initial and final PVTs amounted to approximately 28–30 hours [9].

During each PVT, a light would randomly illuminate every few seconds, prompting the subjects to press a button upon the light appearance, with their reaction times recorded [10]. A lapse, defined as a failure to react or any reaction exceeding 500 milliseconds, was considered a primary outcome measure of PVTs [11].

In safety-critical tasks, the PVT light in DROZY simulates a scenario where an alarm goes off in a control room, indicating a disturbance in the operation process (e.g., temperature and/or pressure rise). Subjects were also required to complete a Karolinska Sleepiness Scale (KSS) form in DROZY, a validated method for subjective sleepiness measurement [12]. The KSS scale, presented in Table 3 [13], allowed subjects to indicate their psychophysical state at the beginning of the PVT. The use of KSS for assessing alertness performance is well-established [14].

3.5. Using DROZY Dataset for model validation

Table 3: KSS scale for self-evaluation of drowsiness

Rating	Verbal descriptions
1	Extremely alert
2	Very alert
3	Alert
4	Fairly alert
5	Neither alert nor sleepy
6	Some signs of sleepiness
7	Sleepy, but no effort to keep alert
8	Sleepy, some effort to keep alert
9	Very sleepy, great effort to keep alert, fighting sleep

Referring to the insights provided in Table 3, it becomes evident that lower Karolinska Sleepiness Scale (KSS) levels align with instances where subjects exhibit optimal performance unaffected by drowsiness. Conversely, higher KSS levels indicate potential states of sleepiness. Acknowledging the subjective nature of KSS as a self-evaluation measure, I adopted a bifurcation into two categories—alert (KSS ≤ 3) and drowsy (KSS ≥ 7). This categorization aims to mitigate potential inconsistencies arising from the inherent difficulty subjects face in precisely determining thresholds between adjacent states (e.g., alert to neutral—classes 4, 5, and 6; neutral to drowsy) [2].

Examining the data, eight subjects self-classified in the initial three alertness stages (levels 1, 2, or 3 in Table 4) during PVT1. Concurrently, DROZY featured ten videos where subjects exerted substantial effort to remain alert, representing the latter three phases of KSS (levels 7, 8, and 9) corresponding to videos during PVT3. The specific KSS classification for each subject is detailed in Table 5.

When assessing the performance of the developed system on the DROZY dataset, each chosen video underwent analysis to determine whether the user received an alert. Accuracy was defined as follows: if a user was categorized as alert and the system did not issue a drowsiness alert, the result was deemed accurate. Conversely, if a user fell into the drowsy category and the system did issue a drowsiness alert, the result was considered accurate. Any deviations from these scenarios were classified as inaccuracies.

3.6. Compute resources needed for the project

The entire project was developed using my personal MacBook Pro (M2 chip). The data collection was done using the computer webcam and the system was developed on Visual Studio Code with Python Programming Language where the libraries dlib, OpenCV, Pandas, and Scikit-Learn were mainly used for implementing the logic.

4. Results

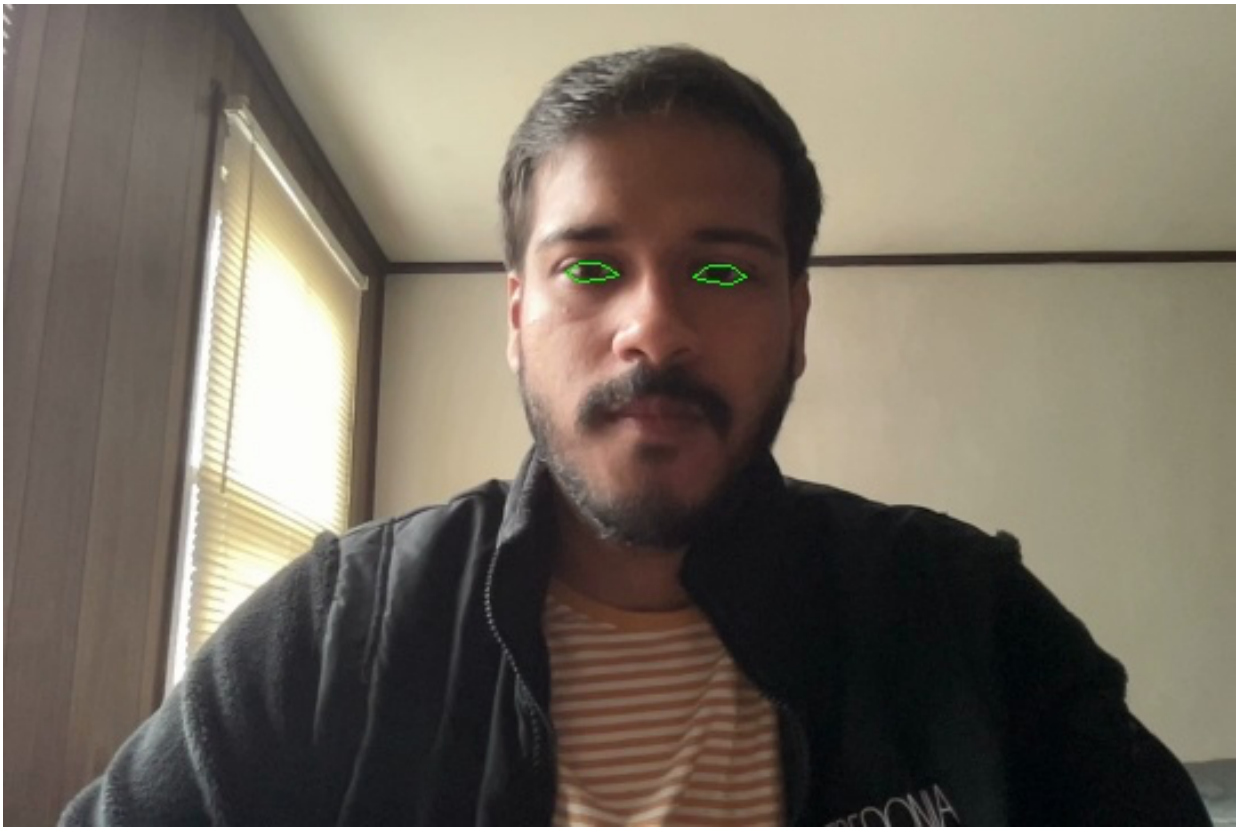


Fig 3: Landmarks generated for eyes using dlib library's facial landmark detector

Table 4: Accuracy performance of SVM models

Model	Test Set				Unseen Data Set			
	0	1	2	Overall	0	1	2	Overall
SVM Linear	0.94	0.86	0.93	0.91	0.95	0.86	0.97	0.92
SVM Radial	0.95	0.88	0.93	0.92	0.92	0.95	0.99	0.95
SVM Radial (Hyperparameter Tuned)	**0.94**	**0.94**	**0.94**	**0.94**	**0.92**	**0.93**	**0.98**	**0.95**

As evident in the table above, the SVM model developed produced an accuracy of 94% on test set and 95% on the unseen data set. This proved that the model was able to generalize well for unseen users.

4.1. Comparison with the state-of-the-art model for the same task:

It can be observed using Table 1 that the model that used a similar approach to mine achieved an accuracy of about 94% on the DROZY dataset. Moreover, the same model's performance on the dataset that they used was also about 94%. This dataset was also created by themselves using a similar experiment to mine. It produced an accuracy of 94.9% on their dataset. This dataset is not available publicly and hence could not be used for comparison. Hence the accuracy values cannot be directly compared to each other. Nevertheless, my model produced 95% accuracy on unseen subjects which proves the feasibility of my model. Moreover, the system has been validated against the DROZY dataset and the results are presented in the next subsection.

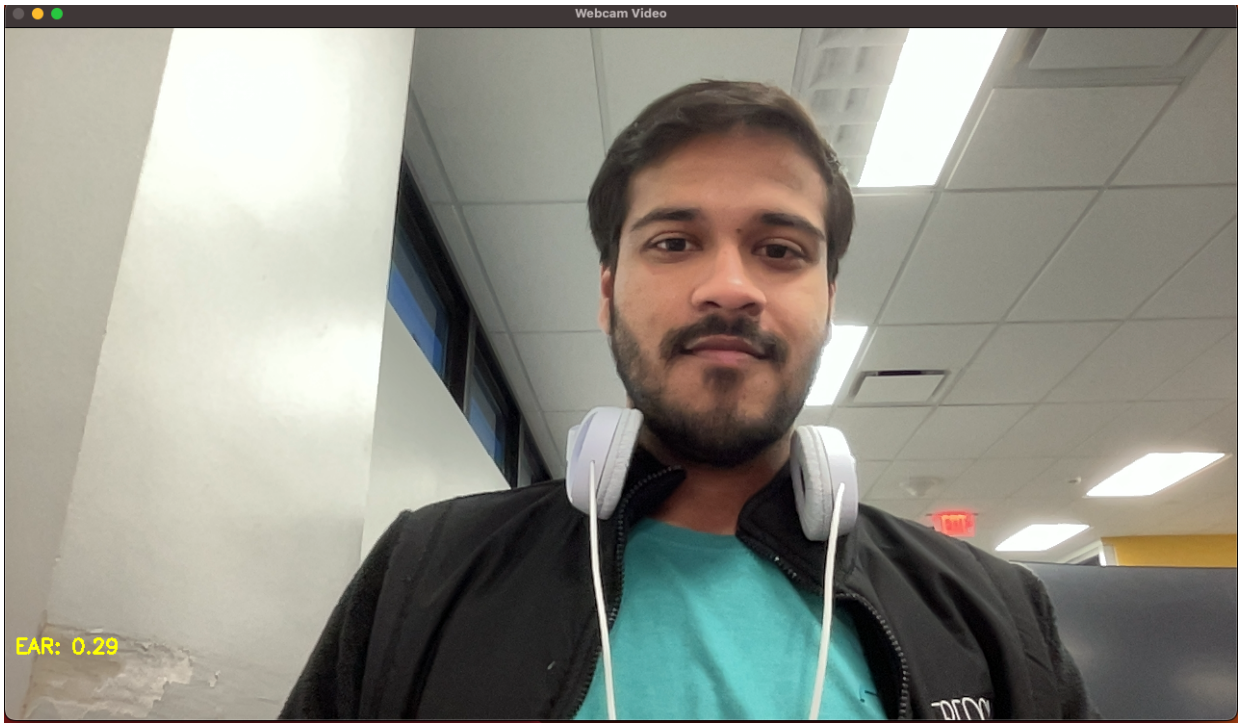


Fig 4: Run time results for alert user

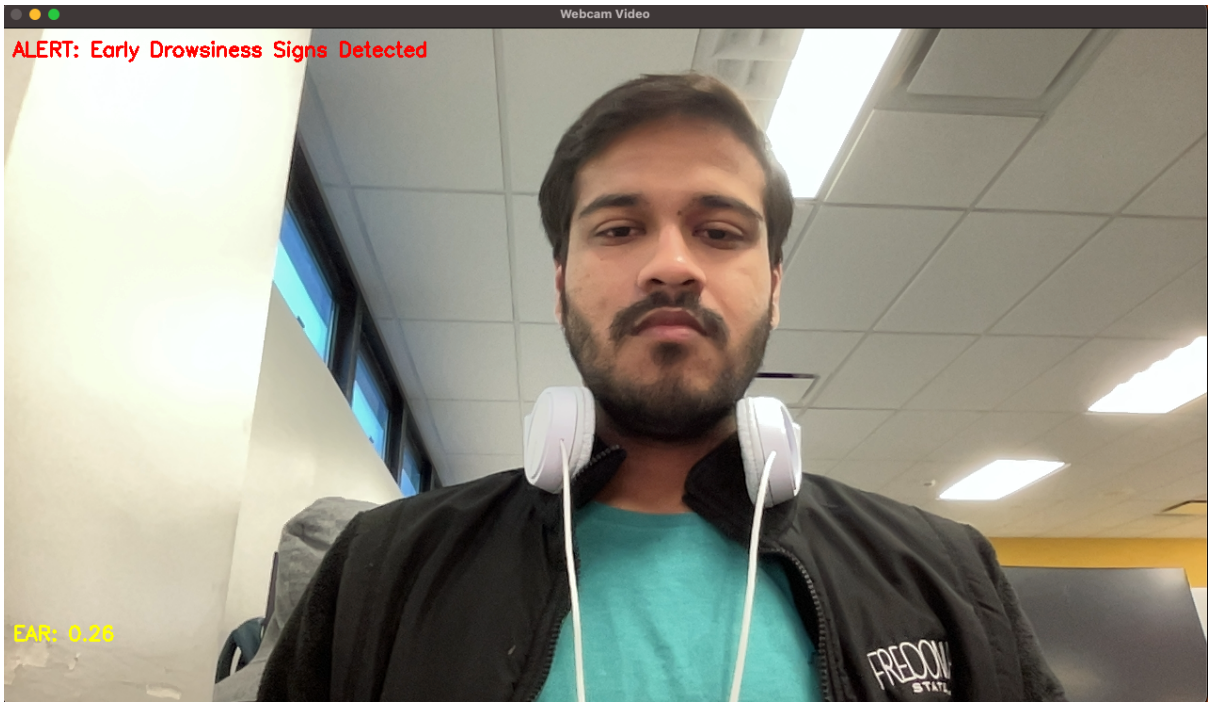


Fig 5: Run time results for user showing early signs of drowsiness

Fig 6: Run time results of user completely drowsy

It was observed on numerous trials with individuals that when the user simulated signs of drowsiness, the model always alerted the user. Examples of this can be seen in the three images above. Hence the model proved be to a successful proof-of-concept. However, the scope of the

experiment was limited due to the time-constraint and some challenges have been discussed in the analysis section.

4.2. Model Validation Results

Table 5: Model validation results

Subject	PVT	KSS	Label Assigned	User warned
1	PVT1	3	Alert	No
2	PVT1	3	Alert	No
3	PVT1	2	Alert	No
5	PVT1	3	Alert	No
6	PVT1	2	Alert	No
8	PVT1	2	Alert	No
10	PVT1	3	Alert	Yes
12	PVT1	2	Alert	No
1	PVT3	7	Drowsy	Yes
4	PVT3	9	Drowsy	Yes
5	PVT3	8	Drowsy	No
6	PVT3	7	Drowsy	Yes
7	PVT3	9	Drowsy	Yes
8	PVT3	8	Drowsy	Yes
9	PVT3	8	Drowsy	Yes
10	PVT3	7	Drowsy	Yes
11	PVT3	7	Drowsy	Yes
14	PVT3	8	Drowsy	Yes

The videos in DROWSY data were captured in very different lighting and demographic conditions than the training data. But the model performed impressively with an accuracy of 88.88%. The logic for calculating the accuracy score has been already explained in Section 3.5. The model performs with close accuracy to the state-of-the-art model that also uses blink patterns as the calculated parameter in Table 1, even though the system developed has been trained on a completely different dataset. The difference in the accuracy is because the system I developed wrongly alerted 2 out of 18 subjects, whereas, the state-of-the-art model wrongly alerted just 1 out of 18 subjects. This proves that this system is not only easy to train and implement but is good at generalizing for various subjects.

5. Analysis

The facial landmarks detection model produced satisfactory results for individuals that do not wear glasses for the most part. However, a few issues were observed with certain individuals.

The first limitation was observed for people with glasses. The below image shows such a user.

Fig 7: Dlib library failing to detect the eyes landmark for person with glasses

The pre-trained model provided by dlib library within Python programming language for detecting the 68 facial landmarks was used for detecting the location of eyes. Upon detection, it was realized that the model performed poorly on individuals who wore corrective glasses. The model might be failing for such users due to these reasons:

1. Reflections and Glare: Glasses often have reflective surfaces that can create highlights or glare. These reflections can confuse the model by altering the appearance of the eyes, making it difficult to accurately detect landmarks.

2. Obstruction: Glasses frames may partially or completely obstruct parts of the eyes. This obstruction can lead to incomplete or inaccurate landmark detection, especially for points located behind the frame.

3. Variability in Frame Types: Glasses come in various shapes, sizes, and styles. The model might not have been trained on a diverse dataset that includes a wide range of glasses types, leading to difficulties in generalizing to novel styles.

4. Differences in Eye Shapes: People have different eye shapes, and glasses may affect the perceived shape of the eyes. The model may not generalize well to individuals with unique eye shapes or those whose eyes are distorted by the glasses.

Hence to complete the proof of concept and prove the feasibility of the core logic, such individuals were excluded from the training data. However, as a future scope, custom eye landmark detection algorithms can be trained with more data of such users so that the model can perform well on such individuals as well.

The second limitation was the failure of dlib.get_frontal_face_detector() to detect faces in frames that had faces in them. This face detector is made using the classic Histogram of Oriented Gradients (HOG) feature combined with a linear classifier, an image pyramid, and sliding window detection scheme. This type of object detector is fairly general and capable of detecting many types of semi-rigid objects in addition to human faces.

However, this model failed to detect the faces of certain individuals. Two such examples are shown below.

Fig 8: Example image 1 where a frontal face exists in the image but dlib's face detector fails to detect the face

Fig 9: Example image 2 where a frontal face clearly exists in the image but dlib's face detector fails to detect the face

The model might be failing for such users due to these reasons:

1. Face Size and Orientation: If the face of an individual is too small or too large in the image, or if it is not facing the camera in a frontal orientation, the detector may struggle to identify the face accurately.

2. Illumination and Shadows: Poor lighting conditions, strong shadows, or uneven illumination can negatively impact face detection. This is because the algorithm relies on patterns of light and dark to identify facial features.

3. Occlusions: If part of the face is occluded by objects such as hands, hair, or accessories, the detector may fail to detect the entire face.

4. Unusual Facial Features: Individuals with unusual facial features, such as extreme facial hair, makeup, or unique facial structures, may pose challenges for the detector, especially if the training data did not adequately cover such variations.

5. Low Resolution: Images with low resolution may not provide enough detail for the detector to accurately identify facial features.

6. Complex Backgrounds: Busy or complex backgrounds can distract the detector, leading to false positives or negatives.

Hence to complete the proof of concept and prove the feasibility of the core logic, such individuals were also excluded from the training data. However, as a future scope, the default face detector in dlib library can be custom-trained and to identify faces in such frames too. Other more sophisticated models like YOLO can also be tried for better accuracy.

Initially, the dataset collected consisted of 24 individuals. However, subjects that fell into any of the above two described categories were removed from the dataset. This resulted in the dataset to be comprised of 14 subjects only. Hence the Subject ID column in Table 2 has values that are not just increasing in step size of 1.

6. Discussion and Lessons Learned

In intricate environments, numerous automated systems necessitate human supervision—a task that, while monotonous, demands a substantial level of attention, particularly when the tasks are crucial for the process and workplace safety. Maintaining adequate alertness among operators becomes imperative in such scenarios to ensure the execution of necessary actions. In this context, I have implemented a methodology for detecting drowsiness based on the eye patterns of individuals observed through video streams. Unlike intrusive biological methods like electrooculogram, I opted for a cost-effective real-time system utilizing computer vision and machine learning (ML) to identify drowsiness using a standard web camera. The approach involves implementing drowsiness rules derived from neuroscience literature, specifically blink patterns, enabling automatic supervision of user alertness to mitigate the risks of human error and prevent accidents. The introduction of a temporal element, achieved by concatenating information from consecutive video frames, coupled with the proficiency of ML models in recognizing various eye behaviors, is a key aspect of the methodology. I explored support vector machines for classification and achieved an accuracy of 95%. Furthermore, the system was validated against on the publicly available DROZY dataset, where it produced an accuracy of 88.88%. This proved the feasibility of using such a system for real-time drowsiness detection. The entire system can be trained and used in real-time on any regular personal laptop without the need for high-speed GPUs. I also realized the challenges of using the pre-trained default models provided by the dlib library and realized the importance of customizing such models for the specific problem at hand for improved efficiency.

Looking ahead to future endeavors within this project, there is an opportunity to enhance the robustness of the model through the implementation of more advanced face detection and facial landmark detection algorithms. These improvements would undoubtedly contribute to a more refined and reliable system. It is noteworthy that throughout this process, I gained a profound appreciation for the efficacy of pre-trained models and their seamless integration into projects, particularly in the context of proof-of-concept endeavors. This realization underscores the value of leveraging existing models to streamline development processes and attain quicker results.

Furthermore, I've come to recognize the pivotal role played by a well-executed feature extraction process. Not only does it facilitate expedited training times, but it also enhances the interpretability of the designed systems. A nuanced understanding of feature extraction can undoubtedly lead to more informed and efficient model training, ultimately influencing the system's performance and effectiveness.

Additionally, a key insight emerged regarding the imperative nature of real-time systems. It is not solely about deploying the most advanced model; rather, equal importance lies in achieving swift execution times. The realization that real-time systems demand not only optimal models, but also efficient code execution has been illuminating. The efficiency of the codebase can exert a significant impact on the runtime speed, thereby influencing the practical usability of the system in real-world scenarios. This underscores the need for a holistic approach that considers both model sophistication and code optimization for the seamless integration of systems into real-world applications.

7. Bibliography

1. Fudail Hasan; Alexey Kashevnik, “State-of-the-Art Analysis of Modern Drowsiness Detection Algorithms Based on Computer Vision”, 25 May 2021
2. Caio Bezerra Souto Maior , Márcio José das Chagas Moura, João Mateus Marques Santana , Isis Didier Lins, “Real-time classification for autonomous drowsiness detection using eye aspect ratio”, 4 May 2020
3. Davis E. King, Dlib-ml: “A Machine Learning Toolkit”, 20 July 2019
4. Kazemi, V., & Sullivan, J. (2014). “One millisecond face alignment with an ensemble of regression trees. In Proceedings of the IEEE computer society conference on computer vision and pattern recognition”
5. Bacivarov, I., Ionita, M., & Corcoran, P. (2008). “Statistical models of appearance for eye tracking and eye-blink detection and measurement. IEEE Transactions on Consumer Electronics”
6. Souto Maior, C. B., Moura, M. C., de Santana, J. M. M., do Nascimento, L. M., Macedo, J. B., Lins, I. D., & Droguett, E. L. (2018). Real-time SVM classification for drowsiness detection using eye aspect ratio. PSAM 2018 – Probabilistic safety assessment and management (September).

7. Stern, J. A., Boyer, D. J., Schroeder, D. J., Touchstone, R. M., & Stoliarov, N. (1996). “Blinks, saccades, and fixation pauses during vigilance task performance: II. Gender and time-of-day”. Washington DC.
8. Caffier, P. P., Erdmann, U., & Ullsperger, P. (2003). Experimental evaluation of eyeblink parameters as a drowsiness measure. European Journal of Applied Physiology.
9. Massoz, Q., Langohr, T., Francois, C., & Verly, J. G. (2016). The ULg multimodality drowsiness database (called DROZY) and examples of use. In 2016 IEEE winter conference on applications of computer vision, WACV 2016.
10. Dinges, D. F., & Powell, J. W. (1985). Microcomputer analyses of performance on a portable, simple visual RT task during sustained operations. Behavior Research Methods, Instruments, & Computers, 17(6), 652–655.
11. Lim, J., & Dinges, D. F. (2008). Sleep deprivation and vigilant attention. Annals of the New York Academy of Sciences.
12. Åkerstedt, T., & Gillberg, M. (1990). Subjective and objective sleepiness in the active individual. International Journal of Neuroscience.
13. Åkerstedt, T., Anund, A., Axelsson, J., & Kecklund, G. (2014). Subjective sleepiness is a sensitive indicator of insufficient sleep and impaired waking function.
14. François, C., Hoyoux, T., Langohr, T., Wertz, J., & Verly, J. G. (2016). Tests of a new drowsiness characterization and monitoring system based on ocular parameters.