ZzzSafe: Catch the Snooze Before You Lose

Driver Drowsiness Detection

Manikanta Varshit [1] Sanika Narmitwar [1] Shruti Chaudhary [1]
Vindhya Jain [1] Yesha Shah[1]

Indian Institute of Technology, Jodhpur
{b22ai038, b22cs046, b22ai037, b22ai060, b22cs067}@iitj.ac.in

Abstract

Driver drowsiness is a critical factor in road accidents, necessitating reliable detection systems for enhanced safety. This project explores computer vision and deep learning approaches to classify drowsiness states using three benchmark datasets: the Driver Drowsiness Dataset (DDD), NTHU Drowsy Driver Detection Dataset, and UTA Real-Life Drowsiness Dataset. We implement both traditional methods (e.g., PERCLOS, eye-blink analysis) and deep learning models, including CNNs for spatial feature extraction and hybrid CNN-LSTM architectures for temporal sequence modeling. We achieved remarkable results with all approaches - XX% for image based, 90% accuracy for video-based models leveraging temporal features, 99% accuracy with a CNN, and 99% for our hybrid CNN-LSTM architecture, which captures spatiotemporal drowsiness cues. This work demonstrates the viability of vision-based drowsiness detection and provides a comparative analysis of methodologies for practical applications. All code files and analyses are available at ZzzSafe.

Contents

1 Introduction

Road safety remains a global challenge, with driver fatigue contributing to nearly 20% of all traffic fatalities. While existing solutions like EEG sensors or steering monitoring exist, vision-based systems offer a non-invasive, scalable alternative for real-time drowsiness detection. This project investigates the intersection of computer vision and deep learning to create a robust, real-time capable system that operates across diverse driving conditions.

Our work systematically evaluates multiple approaches through three key phases: First, we analyze traditional computer vision techniques (facial landmarks, blink-rate analysis, and blur-resistant image processing) on static images. Second, we extend these methods to video streams, incorporating temporal dynamics. Finally, we implement state-of-the-art deep learning architectures, including optimized CNNs and hybrid CNN-LSTM networks, achieving up to 99% classification accuracy on benchmark datasets.

We begin with dataset analysis (Section), progress through methodological comparisons (Section 3), and conclude with practical deployment considerations (Section 4). All implementations are made publicly available to support reproducibility and further development in this critical area of transportation safety.

2 About the Datasets

2.1 Driver Drowsiness Dataset (DDD)

The Driver Drowsiness Dataset (DDD) is a curated collection of facial images specifically designed for training and evaluating driver drowsiness detection systems. It is derived from the Real-Life Drowsiness Dataset, where key video frames were extracted using VLC software and then processed using the Viola-Jones algorithm to isolate the driver's face region. This preprocessing ensures that the dataset is focused solely on relevant facial cues essential for drowsiness analysis.

The dataset consists of over 41,790 RGB images of size 227 × 227 pixels, categorized into two classes: "Drowsy" and "Non-Drowsy". It provides a clean, image-level dataset that simplifies training convolutional neural networks (CNNs) for binary classification tasks.

2.2 NTHU Drowsy Driver Detection Dataset

The NTHU Driver Drowsiness Detection Dataset (NTHU-DDD) is a widely used benchmark for real-time drowsiness detection systems, developed by National Tsing Hua University. It consists of over 36 subjects recorded under varying lighting conditions (daylight, night, and night with infrared) while performing different drowsiness-related behaviors, such as blinking, yawning, nodding, and normal driving.

Each video sequence is annotated with behavior labels like "normal", "yawning", "slow blink", and "nodding", making it ideal for training supervised learning models for driver alertness classification.

While the dataset provides valuable real-world variability, it poses certain challenges. For instance, many subjects in the dataset exhibit subtle facial features such as narrow eye openings or small mouth movements, making it harder for facial landmark-based algorithms to detect drowsiness indicators like eye aspect ratio (EAR) or mouth aspect ratio (MAR). These subtle expressions demand highly precise landmark detection models and robust preprocessing pipelines to ensure accurate feature extraction and classification.

2.3 UTA Real-Life Drowsiness Dataset

The UTA Real-Life Drowsiness Dataset (UTA-RLDD) is a large and realistic dataset created for multi-stage driver drowsiness detection. It captures both visible and subtle signs of drowsiness across various real-life conditions.

- 180 videos (30 hours) from 60 participants
- Three drowsiness levels: Alert, Low Vigilance, and Drowsy (based on KSS)
- Videos captured with both eyes visible, mimicking in-car camera setups
- Diverse participants by age (20–59), ethnicity, gender, with variations like glasses and facial hair
- Frame rate$<$ 30 fps to reflect real-world camera usage

3 Methodologies

3.1 Traditional Computer Vision

Driver drowsiness is a critical factor contributing to road accidents worldwide. Traditional computer vision and machine learning-based approaches have been widely used to detect drowsiness by analyzing facial features, particularly eye and mouth movements. These methods rely on extracting handcrafted features such as eye aspect ratio (EAR), mouth aspect ratio (MAR), blink rate, and head pose variations. By leveraging geometric and statistical techniques, these approaches classify the driver's state as drowsy or alert without requiring large-scale training datasets.

Traditional drowsiness detection can be broadly categorized into:

1. Image-based approaches – Analyze individual frames to compute EAR and MAR for real-time classification.

2. Video-based approaches – Process sequential frames to extract temporal features such as PERCLOS (Percentage of Eye Closure), blink frequency, and eye closure duration for more robust detection.

These methods are computationally efficient and interpretable, making them suitable for real-time embedded systems where deep learning may be too resource-intensive.

3.1.1 Image Based Classification

Blurriness Detection

Blurriness Detection is a rule-based, vision-driven approach to infer driver drowsiness based on the clarity of frames. Drowsiness often leads to prolonged eye closures or unfocused gazes, both of which can cause visual input to become blurry. Additionally, slight involuntary head nods or loss of motor control may result in camera motion, leading to blurred frames. These visual indicators are critical to detect, especially when other cues like EAR or yawns may be ambiguous or occluded.

Figure 1: Class-wise Distribution of Drowsy and Non-Drowsy Image

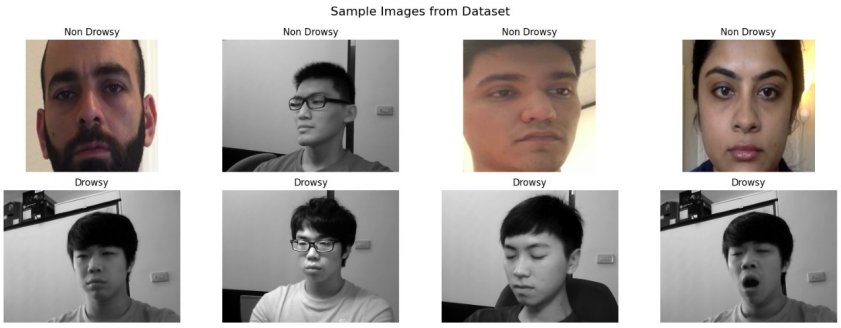


Figure 2: Sample Images Showing Drowsy and Non-Drowsy Subjects.

- Methodology:
  - Laplacian Method: We employed the Variance of the Laplacian method to quantify the blurriness of an image. The Laplacian operator calculates the second derivative of the image, highlighting regions with rapid intensity changes (edges). The variance of this result provides a numeric estimate of the image's sharpness.

    - ∗ High Variance = More edges → Sharp image
    - ∗ Low Variance = Fewer edges → Blurred image
  - Tenengrad Method: The Tenengrad score is computed based on the gradient magnitude derived from the Sobel operator. It emphasizes high-frequency components, which are often reduced in blurry images. This score is sensitive to edge strength and is effective in detecting minor blurring.
  - Sobel Standard Deviation: This metric calculates the standard deviation of the combined horizontal and vertical Sobel gradients. A low standard deviation suggests weak or minimal edge structures, typically associated with blurred regThe system uses empirically derived thresholds to classify each frame as either Drowsy or Alert. A frame is flagged as Drowsy if all three of the following conditions are met:ions.

- Classification Logic:
  The system uses empirically derived thresholds to classify each frame as either Drowsy or Alert. A frame is flagged as Drowsy if all three of the following conditions are met:
  - Laplacian Variance < 100
  - Tenengrad Score < 200
  - Sobel Standard Deviation < 10
- Temporal Smoothing and Decision Aggregation
  - To avoid misclassifications due to transient factors (e.g., camera motion, lighting fluctuations), the system implements temporal smoothing.
  - It maintains a buffer of recent frame predictions and performs majority voting or thresholding over the buffer to determine whether the driver is experiencing sustained drowsiness. This helps reduce noise and increases the system's reliability.

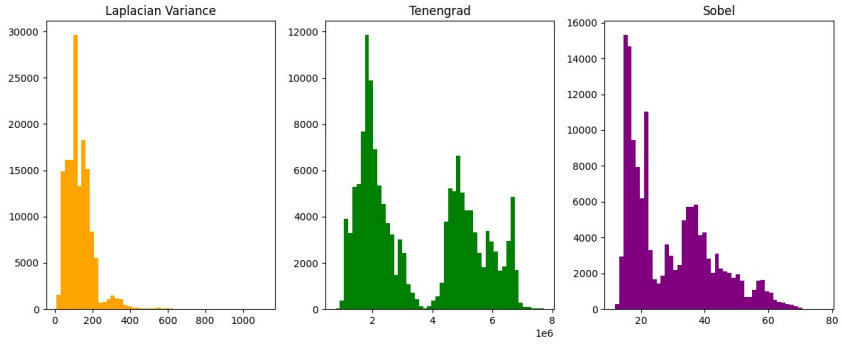


Figure 3: Blurriness Score Distribution Using Laplacian, Tenengrad, and Sobel Metrics

Using Facial Landmarks

In image-based drowsiness detection based on facial landmarks, individual frames from a camera feed are analyzed to determine the driver's state. The key steps involved are:

Facial Landmark Detection: Facial landmarks are detected using libraries such as

- LBF (Local Binary Features) Model
  The LBF (Local Binary Features) model is an efficient facial landmark detection algorithm that combines regression-based approaches with local binary features.

  It starts with a rough face shape estimate, then progressively refines landmark positions through multiple stages. At each stage, it computes simple binary features (comparing pixel intensities around current landmarks) and uses regression trees to predict adjustments. These local predictions are combined into global updates, making the model both fast (due to binary operations) and accurate (through cascaded refinement).

- *Cascaded Regression Framework*
  LBF follows a cascaded regression approach where each stage (or level) progressively refines the landmark positions.
  The model consists of multiple regression stages (typically 4-10) that sequentially improve the landmark estimates.
- *Local Binary Features*
  At each pixel location around the current landmark estimate, the algorithm computes binary features by comparing intensity values between pairs of pixels.
  These binary comparisons are computationally inexpensive yet highly discriminative.

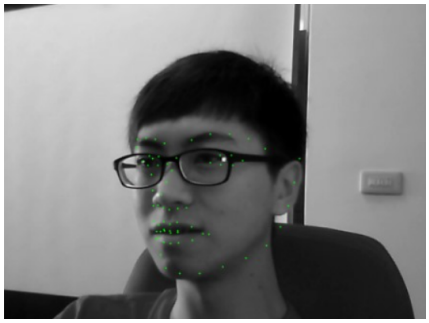


Figure 4: Facial Landmarks detected with LBF Model

- dlib *(shape_predictor_68_face_landmarks.dat)*
  The system employs an Ensemble of Regression Trees (ERT) algorithm with cascaded shape regression, which operates through three key phases:

  1. Face Detection: Utilizes Histogram of Oriented Gradients (HOG) features combined with a linear SVM classifier and generates bounding boxes for facial regions of interest

  2. Landmark Localization: The model implements a cascaded regression framework comprising 500 refinement stages, where each stage consists of an ensemble of gradient-boosted regression trees. At every stage, the model processes pixel-intensity difference features centered around the current estimates of facial landmarks. This iterative approach allows the model to progressively refine the landmark positions, significantly reducing prediction error from the initial mean shape.

  3. Output Generation: The model outputs 68 predefined facial landmarks that adhere to the iBUG standard, with points organized as follows: 17 for the jawline, 10 for the eyebrows, 9 for the nose, 12 for the eyes, and 20 for the mouth. This structured arrangement facilitates precise facial feature localization.

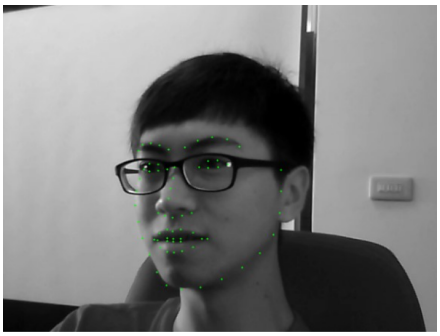


Figure 5: All Facial Landmarks with dlib

Figure 6: Target Facial Landmarks

- MediaPipe
  We used MediaPipe's face landmark detection for **real-time** analysis, which utilizes an optimized machine learning pipeline with gradient boosting and regression trees.

The solution provides 468 facial landmarks through an efficient algorithmic approach designed specifically for edge devices.

Unlike CNN-based methods, this implementation achieves real-time performance without requiring GPU acceleration, making it ideal for our drowsiness detection system.

For our drowsiness detection, we specifically utilized landmark subsets for eye contours (points 33-46 and 246-249) and mouth boundaries (points 61-91).

Figure 7: All Facial Landmarks with Mediapipe

Figure 8: Target Facial Landmarks

Feature Extraction: The primary features extracted are

- Eye Aspect Ratio (EAR): Measures eye openness using the ratio of vertical to horizontal eye landmarks.

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

  where:

  * $p_1, p_2, \ldots, p_6$ are the facial landmarks around the eye
  * $||\cdot||$ denotes the Euclidean distance between two points.

  When the eye is open, EAR remains relatively constant.
  When the eye closes, EAR drops significantly (approaching zero).

- Mouth Aspect Ratio (MAR): Detects yawning by measuring mouth openness.

$$\text{MAR} = \frac{||m_2 - m_8|| + ||m_3 - m_7|| + ||m_4 - m_6||}{3 \times ||m_1 - m_5||}$$

  where:

  * $m_1, m_2, \ldots, m_8$ are the facial landmarks around the mouth

  A high MAR indicates mouth opening (yawning).
  A threshold (e.g., 0.75) is used to classify yawning.
  Frequent yawning is a strong indicator of drowsiness.

Classification: A threshold-based approach is used

- If EAR < 0.22 for consecutive frames $\rightarrow$ drowsy.
- If MAR $> 0.6 \rightarrow$ possible yawning.

This method is fast and suitable for real-time applications but may lack robustness under varying lighting and head poses.

Results and Takeaways
We achieved promising results using both static and real-time detection methods. However, the current approach analyzes individual frames without considering temporal patterns, which limits

its ability to detect gradual drowsiness progression. Future improvements should incorporate time-based analysis for more robust detection.

Figure 9: LBF Model Result

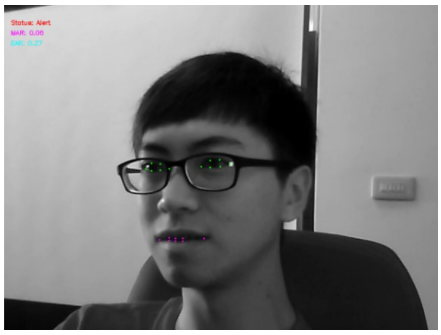


Figure 10: dlib result

While both were able to detect correct status, dlib performs better at highlighting facial landmarks.

3.1.2 Video Based Classification

This approach leverages handcrafted physiological features derived from facial landmarks to classify driver drowsiness. Using video frames present in two different datasets, two primary features—**Eye Aspect Ratio (EAR)** and **Mouth Aspect Ratio (MAR)**—are computed for each frame to reflect eye closure and mouth openness, respectively. These features are then aggregated to derive higher-level statistical indicators that can signify drowsy behavior.

Methodology:

We begin by detecting facial landmarks using the pre-trained dlib model, which identifies 68 specific facial points. From these, we extract the eye and mouth regions and compute EAR and MAR using Euclidean distance-based geometric formulations. For each group of video frames (associated with a specific subject or video clip), we extract a set of temporal features:

- **PERCLOS**: Percentage of frames where EAR less than threshold (a strong indicator of drowsiness).
- **MCD (Mean Closure Duration)**: Average EAR during eye-closed frames.
- **AOL (Average Open Level)**: Average EAR during eye-open frames.
- **BF (Blink Frequency)**: Number of blinks per frame.
- **OV (Open Variability)**: Standard deviation of EAR in open-eye frames.
- **CV (Closed Variability)**: Standard deviation of EAR in closed-eye frames.
- **MAR Mean and Std**: Mean and standard deviation of MAR, capturing yawning or mouth movement patterns.

These features are compiled into a structured dataset. The target labels ("Drowsy" or "Non Drowsy") are encoded using `LabelEncoder`. The dataset is then split into training and testing sets and models are trained.

Results:

DDD Dataset
1. Random Forest

- **Accuracy**: 0.9
- **Precision**: 0.92

- **Recall**: 0.9
- **F1-Score**: 0.9

2. Support Vector Machine (SVM)

- **Accuracy**: 0.9
- **Precision**: 0.91
- **Recall**: 0.9
- **F1-Score**: 0.9

NTHU DDD Dataset

1. Fisher's Linear Discriminant Analysis (LDA)

The main objective of LDA is to maximize the separability between classes while minimizing the variance within each class. This is achieved by projecting the feature vectors onto a line defined by a set of coefficients (also known as canonical discriminant coefficients) such that the ratio of between-class variance to within-class variance is maximized.

The resulting coefficients w serve as weights for each input metric and indicate their contribution toward distinguishing drowsiness levels.

Table 1: Coefficients for Drowsiness Detection Features

	PERCLOS	**MCD**	**AOL**	**BF**	**OV**	**CV**	**Const.**
Fn 0	-0.006	-0.319	0.159	-0.099	-0.781	-0.056	-1.011
Fn 1	0.005	0.273	-0.136	0.085	0.669	0.048	-0.793

Accuracy: 0.71

2. Support Vector Machine (SVM)

- **Accuracy**: 0.77
- **Precision**: 0.84
- **Recall)**: 0.77
- **F1-Score**: 0.75

3. Decision Tree

- **Accuracy**: 0.85
- **Precision**: 0.85
- **Recall)**: 0.85
- **F1-Score**: 0.85

3.2 Deep Learning Based

3.2.1 Convolutional Neural Network (CNN)

This section presents a PyTorch-based deep learning framework for drowsiness detection using a custom Convolutional Neural Network (CNN). The model processes individual video frames, extracting spatial features to classify driver states.

- Dataset Preparation
  - Preprocessing
    - For each video in the dataset, frames are sampled at a fixed frame interval. This reduces redundancy and ensures temporal coverage.
    - Resized all frames to 112×112 pixels.
    - Converts BGR→RGB and normalizes to [0,1] for PyTorch

* Data Splitting
  - Split the dataset into 70% training, 15% validation, 15% test

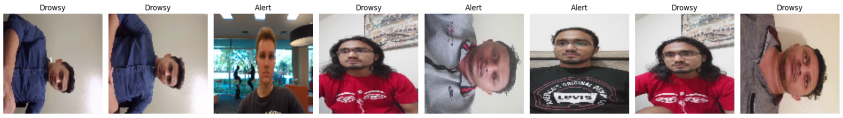
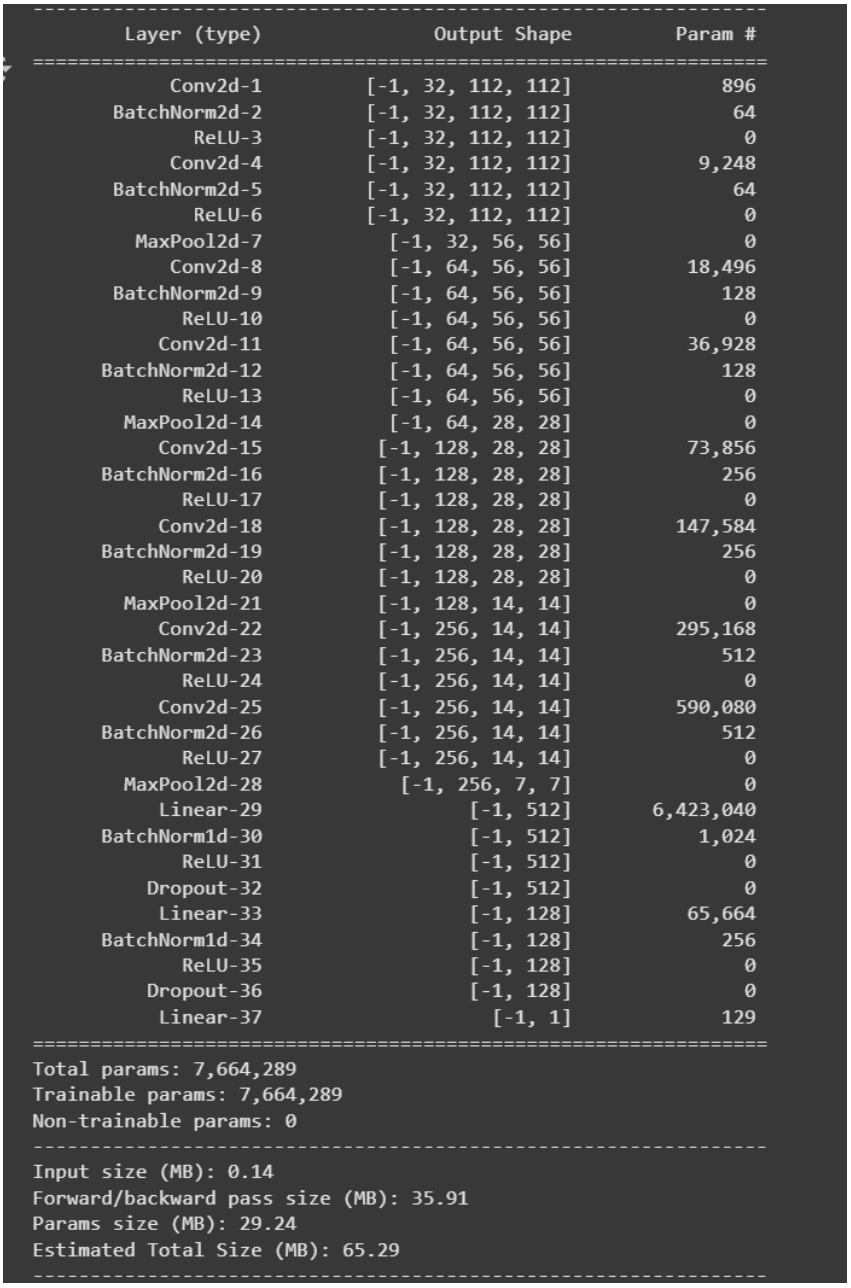


Figure 11: Training examples with labels.

– Model Architecture

To classify each video frame as either drowsy or alert, a deep Convolutional Neural Network (CNN) model was developed. This architecture was designed to progressively learn visual features using stacked convolutional layers followed by fully connected layers for binary classification.

* Key Design Elements:
  - Convolutional Blocks: The model consists of four convolutional blocks. Each block contains two convolutional layers followed by batch normalization and ReLU activation.
  - Weight Initialization: The model uses Kaiming Normal initialization for convolutional layers and standard normal initialization for linear layers, with batch normalization layers initialized to identity.



Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 32, 112, 112]	896
BatchNorm2d-2	[-1, 32, 112, 112]	64
ReLU-3	[-1, 32, 112, 112]	0
Conv2d-4	[-1, 32, 112, 112]	9,248
BatchNorm2d-5	[-1, 32, 112, 112]	64
ReLU-6	[-1, 32, 112, 112]	0
MaxPool2d-7	[-1, 32, 56, 56]	0
Conv2d-8	[-1, 64, 56, 56]	18,496
BatchNorm2d-9	[-1, 64, 56, 56]	128
ReLU-10	[-1, 64, 56, 56]	0
Conv2d-11	[-1, 64, 56, 56]	36,928
BatchNorm2d-12	[-1, 64, 56, 56]	128
ReLU-13	[-1, 64, 56, 56]	0
MaxPool2d-14	[-1, 64, 28, 28]	0
Conv2d-15	[-1, 128, 28, 28]	73,856
BatchNorm2d-16	[-1, 128, 28, 28]	256
ReLU-17	[-1, 128, 28, 28]	0
Conv2d-18	[-1, 128, 28, 28]	147,584
BatchNorm2d-19	[-1, 128, 28, 28]	256
ReLU-20	[-1, 128, 28, 28]	0
MaxPool2d-21	[-1, 128, 14, 14]	0
Conv2d-22	[-1, 256, 14, 14]	295,168
BatchNorm2d-23	[-1, 256, 14, 14]	512
ReLU-24	[-1, 256, 14, 14]	0
Conv2d-25	[-1, 256, 14, 14]	590,080
BatchNorm2d-26	[-1, 256, 14, 14]	512
ReLU-27	[-1, 256, 14, 14]	0
MaxPool2d-28	[-1, 256, 7, 7]	0
Linear-29	[-1, 512]	6,423,040
BatchNorm1d-30	[-1, 512]	1,024
ReLU-31	[-1, 512]	0
Dropout-32	[-1, 512]	0
Linear-33	[-1, 128]	65,664
BatchNorm1d-34	[-1, 128]	256
ReLU-35	[-1, 128]	0
Dropout-36	[-1, 128]	0
Linear-37	[-1, 1]	129

Total params: 7,664,289
Trainable params: 7,664,289
Non-trainable params: 0

Input size (MB): 0.14
Forward/backward pass size (MB): 35.91
Params size (MB): 29.24
Estimated Total Size (MB): 65.29

Figure 12: CNN Model Architecture

- Training Configuration
  - Loss Function: Binary Cross-Entropy (BCELoss)
  - Optimizer: Adam
    - Learning rate: 1e-4
    - Adaptive momentum for stable convergence
- Results
  The trained model was evaluated on the test dataset using several standard metrics. It achieved:
  - Accuracy : 1.0
  - Precision : 1.0
  - Recall/Sensitivity : 1.0
  - Specificity : 1.0
  - F1-Score : 1.0

```
Confusion Matrix:
True Negatives: 1324 (Awake predicted as Awake)
False Positives: 0 (Awake predicted as Drowsy)
False Negatives: 0 (Drowsy predicted as Awake)
True Positives: 1251 (Drowsy predicted as Drowsy)

Classification Report:
              precision    recall  f1-score   support

       Awake       1.00      1.00      1.00      1324
      Drowsy       1.00      1.00      1.00      1251

    accuracy                           1.00      2575
   macro avg       1.00      1.00      1.00      2575
weighted avg       1.00      1.00      1.00      2575
```

Figure 13: Classification Report.

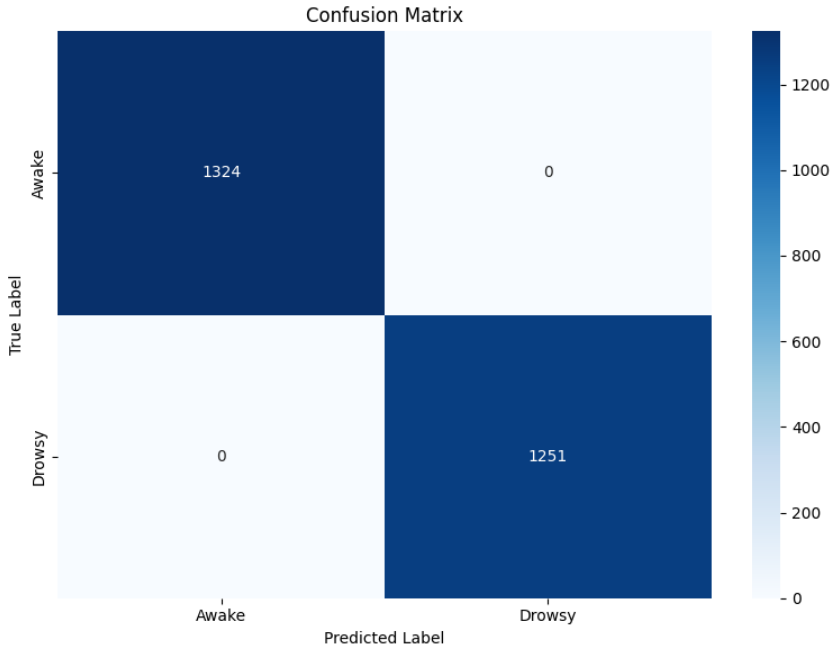


Figure 14: Confusion Matrix

Figure 15: CNN Model Results

3.2.2 CNN + LSTM Hybrid Model

This approach leverages spatial feature extraction through CNN layers and temporal modeling via LSTM units to detect driver drowsiness from video inputs. The system is trained on sliding window segments of video frames and evaluated on multiple metrics such as accuracy, precision, recall, and F1 score. The results demonstrate the effectiveness of the hybrid model in detecting drowsy states compared to alert ones, offering a promising solution for driver monitoring systems.

Sliding Window Approach

We used **UTA Real-Life Drowsiness Dataset** for this approach since it deals with videos which are sequential data. A custom dataset class, `SlidingWindowDrowsinessDataset`, is implemented to handle video data for drowsiness detection. This class performs the following tasks:

[leftmargin=*, label=–]**Scanning the directory:** Recursively searches for video files with valid extensions (`.mov`, `.mp4`) and assigns a label by parsing the filename. **Frame extraction using sliding windows:** A sliding window with a fixed number of frames (e.g., 16) and a specified stride is used to capture sequential information. Each window forms an input sample to the model.

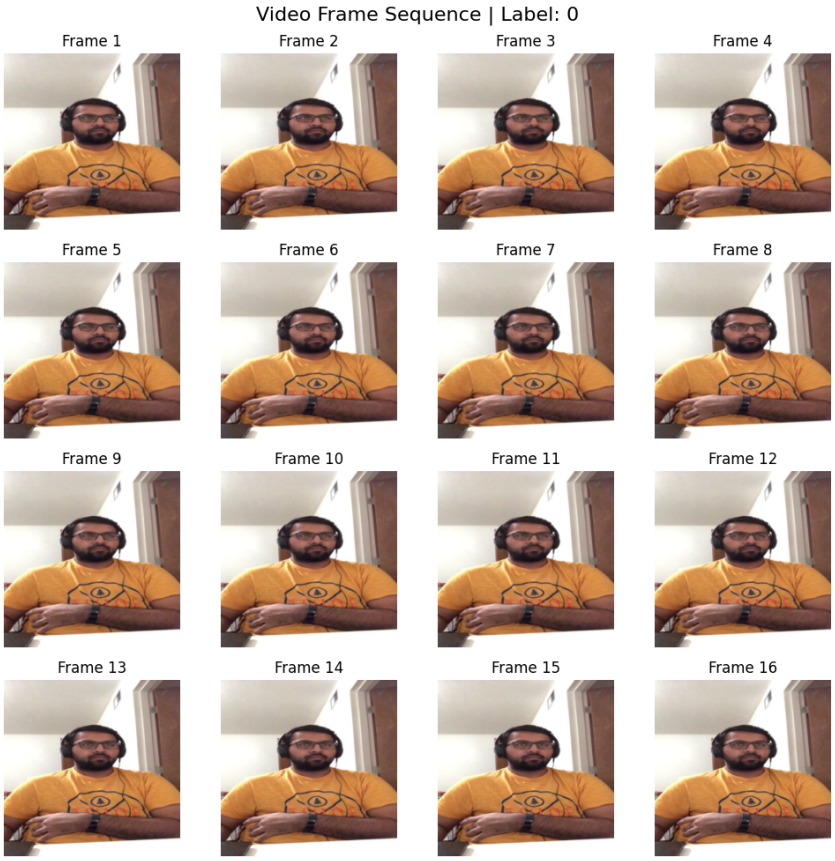


Figure 16: Example input to the CNN+LSTM Mode

Model Architecture

The proposed architecture consists of two key modules designed to extract spatial features and model temporal dependencies.

CNN Feature Extractor: The CNN feature extractor is responsible for obtaining robust spatial features from each video frame through the following components:

[leftmargin=*, label=–]**Layers:** Multiple convolutional layers are stacked to progressively extract detailed spatial features. **Activation and Pooling:** Each convolutional layer is followed by a ReLU activation function and a pooling operation to reduce the spatial dimensions.

LSTM for Temporal Modeling: The LSTM module models the temporal dynamics in the sequence of features extracted by the CNN:

[leftmargin=*, label=–]**Sequential Modeling:** An LSTM processes the sequence of spatial features extracted from the individual frames, effectively capturing temporal dependencies. **Final Classification:** The output from the last time step of the LSTM is passed through a fully connected layer to predict drowsiness.

Training Strategy

The network is trained using the following configurations:

[leftmargin=*, label=–]**Loss Function:** Cross-entropy loss is employed for classification. **Optimizer:** The Adam optimizer is used with an initial learning rate of 0.0001. **Epochs:** The model is trained for a specified number of epochs (e.g., 10).

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
CNN_LSTM                                 [8, 2]                    --
├─Sequential: 1-1                        [128, 512, 1, 1]          --
│    └─Conv2d: 2-1                       [128, 64, 224, 224]       1,792
│    └─ReLU: 2-2                         [128, 64, 224, 224]       --
│    └─MaxPool2d: 2-3                    [128, 64, 112, 112]       --
│    └─Conv2d: 2-4                       [128, 128, 112, 112]      73,856
│    └─ReLU: 2-5                         [128, 128, 112, 112]      --
│    └─MaxPool2d: 2-6                    [128, 128, 56, 56]        --
│    └─Conv2d: 2-7                       [128, 256, 56, 56]        295,168
│    └─ReLU: 2-8                         [128, 256, 56, 56]        --
│    └─MaxPool2d: 2-9                    [128, 256, 28, 28]        --
│    └─Conv2d: 2-10                      [128, 512, 28, 28]        1,180,160
│    └─ReLU: 2-11                        [128, 512, 28, 28]        --
│    └─AdaptiveAvgPool2d: 2-12           [128, 512, 1, 1]          --
├─LSTM: 1-2                              [8, 16, 256]              1,314,816
├─Linear: 1-3                            [8, 2]                    514
==========================================================================================
Total params: 2,866,306
Trainable params: 2,866,306
Non-trainable params: 0
Total mult-adds (Units.GIGABYTES): 367.18
==========================================================================================
Input size (MB): 77.07
Forward/backward pass size (MB): 6165.89
Params size (MB): 11.47
Estimated Total Size (MB): 6254.42
==========================================================================================
```

Figure 17: Model Architecture

Results/Outputs:

Visualizations of model predictions on test samples show that the spatial features extracted by the CNN are robust in capturing the necessary details from individual frames and the LSTM effectively models temporal dynamics, leading to improved classification of sequences where drowsiness cues evolve over time.

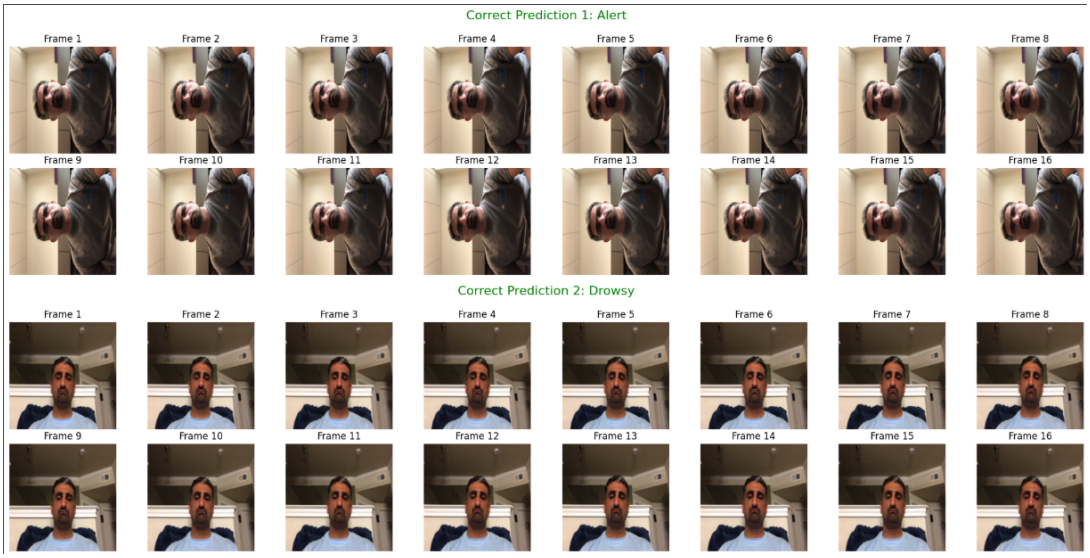


Figure 18: CNN+LSTM Predictions

Classification Report:

	precision	recall	f1-score	support
Alert	1.00	1.00	1.00	1056
Drowsy	1.00	1.00	1.00	860
accuracy			1.00	1916
macro avg	1.00	1.00	1.00	1916
weighted avg	1.00	1.00	1.00	1916

Figure 19: CNN+LSTM Classification Report

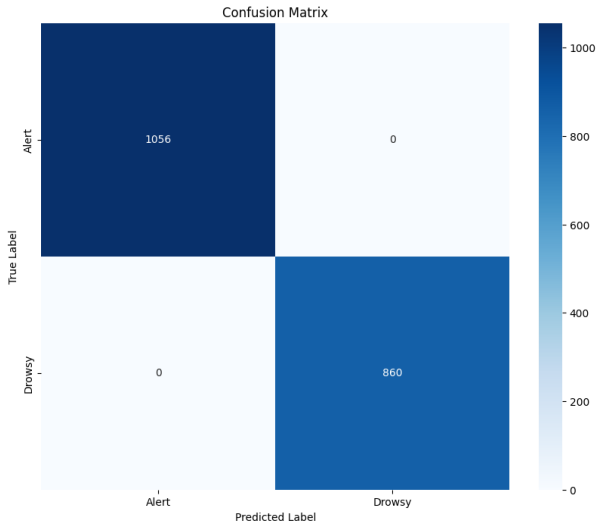


Figure 20: CNN+LSTM Confusion Matrix

Challenges :

- **Real-Time Constraints:** While the model is effective in batch testing, deployment in real-time systems isn't very effecient.
- **Diverse Dataset:** We need a more diverse dataset since for the model to generalise well.

4 GitHub Repository

The complete implementation, including dataset preprocessing scripts, model training code, and deployment, is available in our public GitHub repository:

https://github.com/legend4137/ZzzSafe

5 Deployment and Interface

To demonstrate the practical utility of our drowsiness detection system, we deployed the final model as a user-friendly application. The interface captures live video feed, performs real-time inference

using the trained model, and alerts the user upon detecting signs of drowsiness.
Below are sample screenshots of the deployed application, showcasing real-time drowsiness alerts.

Link to App : https://cv-drowsiness-detection.streamlit.app/

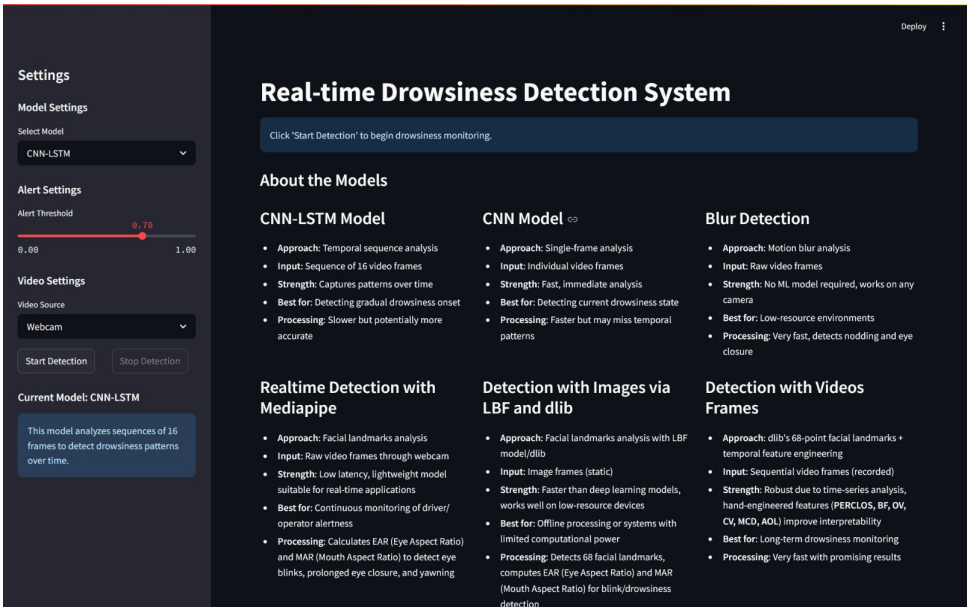


Figure 21: App Home Screen

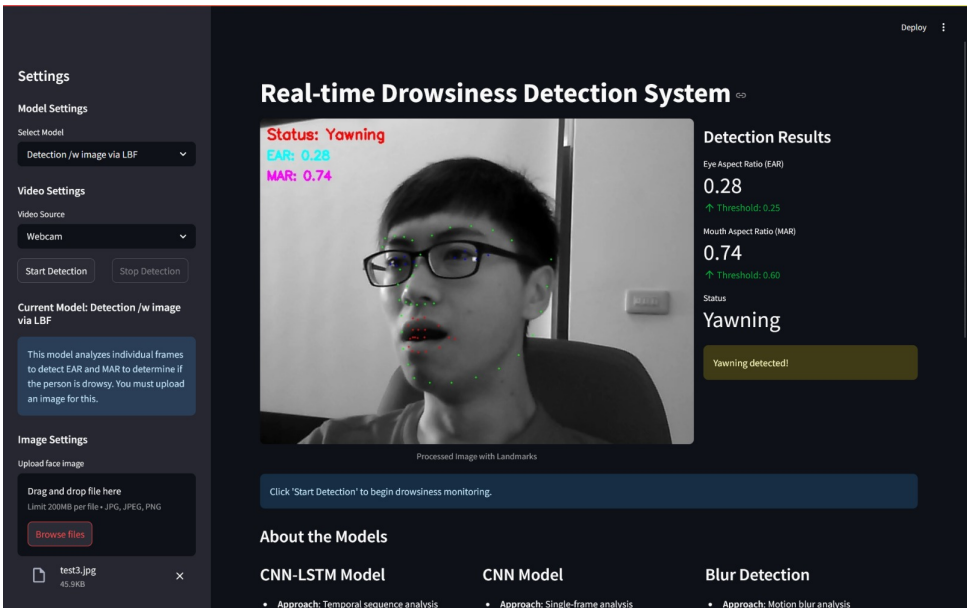


Figure 22: Image Prediction

6 Contributions

The project was collaboratively completed by a team of five members. The individual contributions are as follows:

– **Yesha Shah (B22CS067)**: Image-based drowsiness detection using blur estimation metrics (Laplacian Variance, Tenengrad, Sobel) to quantify eye sharpness. Performed statistical analysis of blur distributions across drowsy and non-drowsy classes using NTHUDDD2 and DDD

datasets. Developed a real-time webcam-based detection system leveraging blur metrics for live drowsiness assessment.

- **Sanika Narmitwar (B22CS046)**: Image based drowsiness detection using Dlib, video based detection with traditional CV (DDD dataset).
- **Vindhya Jain (B22AI060)**: Image based drowsiness detection using LBF, real-time drowsiness detection with Mediapipe, video based detection with traditional CV (nthuddd dataset).
- **Shruti Chaudhary (B22AI037)**: Implemented a CNN-based approach to detect driver drowsiness directly from facial images. The model learns features automatically, removing the need for handcrafted metrics like EAR and MAR.
- **Manikanta Varshit (B22AI038)**: Developed a CNN+LSTM model to capture both spatial and temporal patterns in facial video frames for drowsiness detection. The CNN extracts frame-wise features, while the LSTM models temporal dependencies across sequences. This approach enhances detection accuracy by incorporating motion and behavioral cues over time.

7 Conclusion

ZzzSafe presents a multi-faceted solution for detecting driver drowsiness using modern deep learning and rule-based vision techniques. By combining CNN-based models for image/video analysis and lightweight real-time facial feature extraction techniques, our system ensures a broad and reliable drowsiness detection capability. The deployed web app makes it easy to test the models and monitor driver alertness through various indicators. With future improvements, the system can be expanded to include real-time webcam alerts, mobile integration, and in-vehicle use cases.

References

[1] DataSets:
https://www.kaggle.com/datasets/ismailnasri20/driver-drowsiness-dataset-ddd
https://www.kaggle.com/datasets/faisal7/nthuddd
https://www.kaggle.com/datasets/rishab260/uta-reallife-drowsiness-dataset

[2] Others:
https://www.kaggle.com/code/esraameslamsayed/driver-drowsiness-detection-cnn-mobilenetv2
www.kaggle.com/code/rowydaelshaer/driver-drowsiness-detection-cnn-lstm
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6216768&tag=1
https://www.hackersrealm.net/post/real-time-driver-drowsiness-detection-opencv
https://github.com/z-mahmud22/Dlib_Windows_Python3.x
https://www.mdpi.com/1424-8220/25/3/812