**Multimodal Driver Drowsiness Detection System**

Developed by: - Priyansh Singh

Mentor: - Mrs. Surabhi Purwar

**Implementation of the Multimodal Driver Drowsiness Detection System**

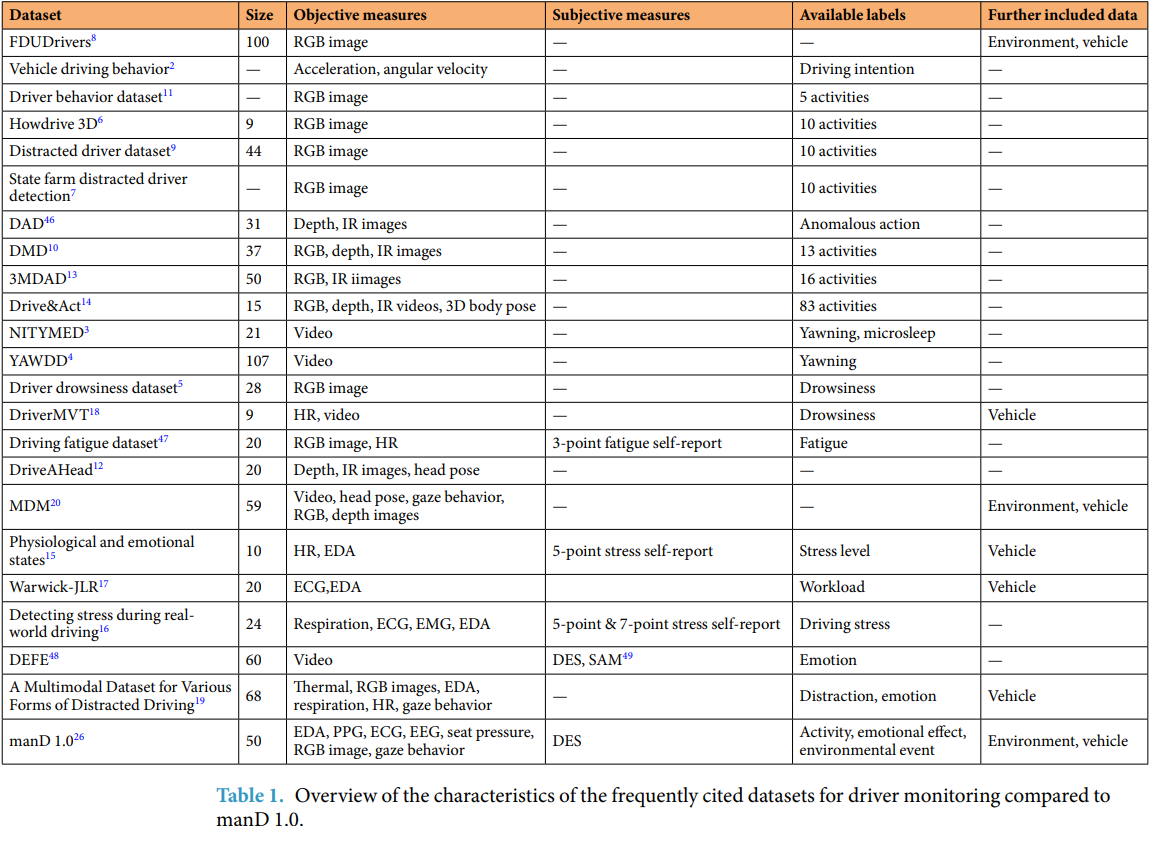
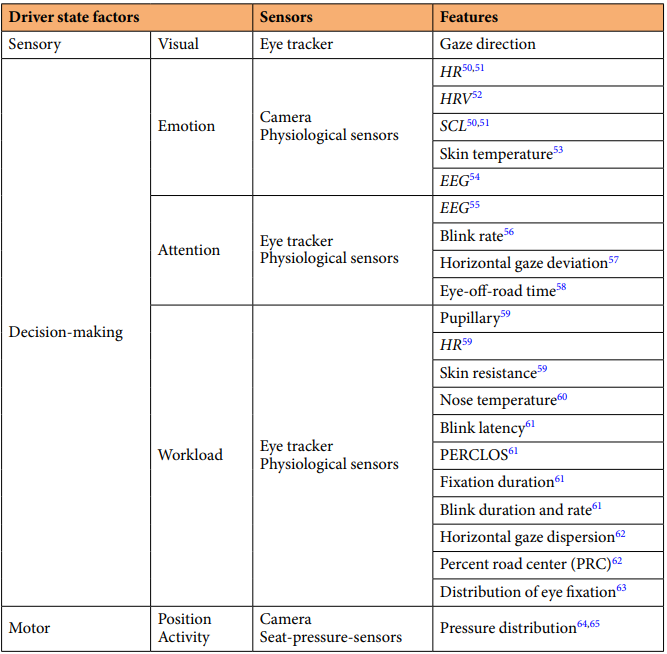
**1. Introduction**

Drowsy driving is a significant cause of road accidents and fatalities worldwide. It reduces a driver's alertness, reaction time, and decision-making ability, leading to potentially fatal consequences. Traditional drowsiness detection methods, such as monitoring eye closure or steering patterns, often fail due to external factors like lighting conditions, occlusions, and variations in driver behaviour. To enhance the reliability of drowsiness detection, we propose a **multimodal approach** that integrates multiple data sources, ensuring a more comprehensive and accurate analysis of driver alertness.

**2. Data Collection and Preprocessing**

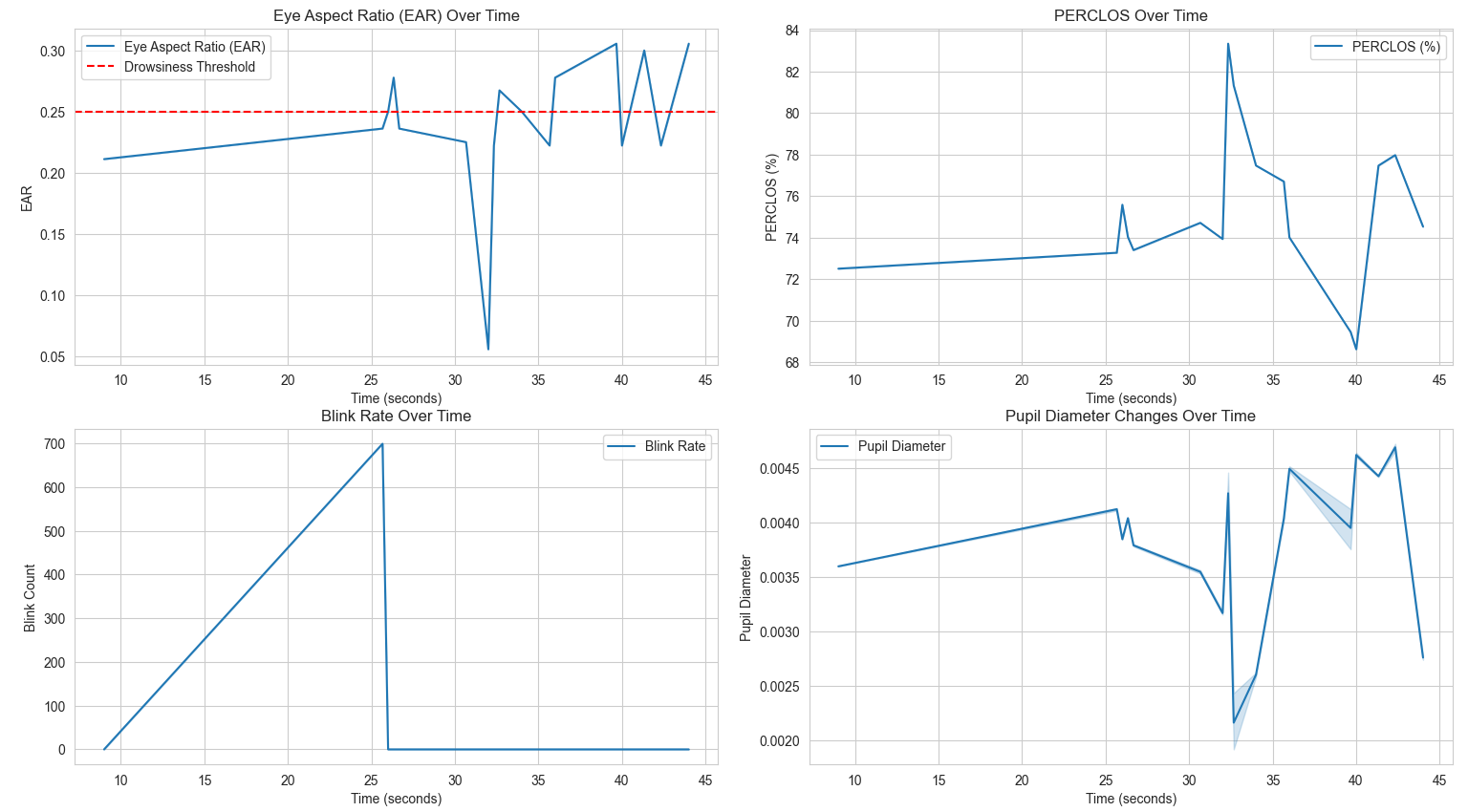
**2.1 Data Collection and Dataset Details**

The dataset used in this project is sourced from the **Harvard Dataverse** and can be accessed at [this link](https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/SG9TMD). The dataset consists of **50 participants**, each contributing **various multimodal data sources**, including:

* **Features:**
  + **Visual Data:** Facial expressions (anger, sadness, fear, etc.), eye closure, yawning, and head position.
  + **Physiological Data:** EEG, ECG, PPG, and EDA signals.
  + **Behavioural Data:** Steering patterns, heart rate variability, and speech patterns.
  + **Environmental & Vehicle State Data:** Speed, acceleration, lane deviations, and road conditions.
* Code file:- m1.py

**2.2 Feature Extraction & Preprocessing**

* Identified and extracted relevant features required for drowsiness detection.
* Located data points necessary for analysis.
* Applied preprocessing techniques such as noise reduction, normalization, and synchronization of multimodal data sources.
* Code file:- visual\_of\_m1.py



**2.3 Video Processing & Labelling**

1. Video Processing & Labelling

* Collected and labeled videos based on drowsiness indicators.
* Extracted and categorized frames into **Drowsy** and **Alert** sets.
* Synchronized frames with eye-tracking data.

2. Eye-Tracking Data Processing

* Processed and cleaned eye-tracking data, filling missing values.
* Interpolated data to align with video frames.

3. Face & Eye Detection

* Detected faces and extracted eye landmarks using **Dlib**.
* Computed **Eye Aspect Ratio (EAR)** for drowsiness detection.

4. Feature Extraction & Labelling

* Extracted **PERCLOS**, **blink rate**, **EAR**, and **pupil diameter**.
* Labeled frames as **Drowsy** or **Alert** based on thresholds.

5. Data Saving & Export

* Merged labeled frames with eye-tracking data.
* Saved processed **CSV files** and labeled images.

📂 **Code file:** frames\_with\_labels2.py

**3. Model Training and Implementation**

**3.1 CNN Model for Visual Analysis**

1. Trained a **Convolutional Neural Network (CNN)** on labelled frames (drowsy vs. alert).
2. Integrated **Deep Face** for facial emotion recognition (anger, sadness, fear).
3. Applied OpenCV and Dlib for **face detection and landmark tracking**.
4. Successfully generated the CNN model.
5. Tested the CNN model for drowsiness detection with **audio alerts** when drowsiness is detected.

**Implementation Details:**

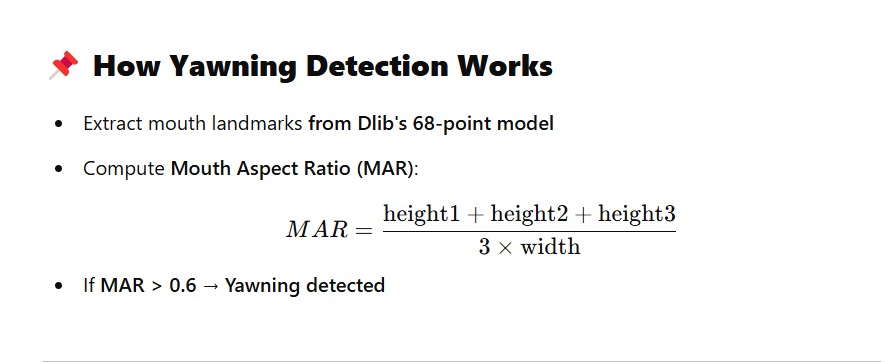
1. **Data Loading & Preprocessing:**

* Loaded image data from directories named Face\_cropped1 to Face\_cropped27 within the extracted\_frames folder.
* Each directory contained subfolders named "alert" and "drowsy," with images representing corresponding states.
* Resized images to 64x64 pixels and converted them to grayscale.
* Normalized pixel values to the range [0, 1].
* Assigned numerical labels: 0 for "alert" and 1 for "drowsy."
* Created lists X (images) and y (labels).
* Converted X and y to NumPy arrays.
* Reshaped X to the format expected by the CNN (samples, height, width, channels).
* Split the dataset into training and validation sets (80/20 split, random state 42).

1. **Model Training:**

* Defined a CNN model using TensorFlow/Keras.
  + Model architecture:
    - Convolutional layer with 32 filters, 3x3 kernel, ReLU activation, and input shape (64, 64, 1).
    - Max pooling layer (2x2).
    - Convolutional layer with 64 filters, 3x3 kernel, and ReLU activation.
    - Max pooling layer (2x2).
    - Flatten layer.
    - Dense layer with 128 neurons and ReLU activation.
    - Dropout layer (0.5).
    - Dense output layer with 2 neurons and softmax activation (for binary classification).
* Compiled the model using the Adam optimizer, sparse categorical cross-entropy loss function, and accuracy metric.
* Trained the model for 50 epochs with a batch size of 16, using the validation set for monitoring performance.

1. **Model Saving & Real-Time Testing:**

* Saved the trained CNN model as drowsiness\_cnn\_model.h5.
* **Real-time testing:**
  + Loaded the saved drowsiness\_cnn\_model.h5 model.
  + Initialized Dlib's face detector.
  + Opened a webcam capture using OpenCV.
  + Entered a loop to continuously capture and process video frames.
  + Converted each frame to grayscale.
  + Detected faces in the grayscale frame using Dlib's face detector.
  + For each detected face:
    - Cropped the face region from the grayscale frame.
    - Preprocessed the face image by resizing it to 64x64 pixels and normalizing pixel values to the range [0, 1].
    - Reshaped the preprocessed image to the input format expected by the CNN model (1, 64, 64, 1).
    - Used the CNN model to predict drowsiness (0 = Drowsy, 1 = Alert).
    - Displayed the drowsiness label and a bounding box around the detected face on the video frame.
  + Displayed the processed video frame with annotations.
  + Exited the loop when the user presses the 'q' key.
  + Released the webcam capture and closed all OpenCV windows.
  + This verified the functionality of the model in a live environment.

1. **Code file:- CNN\_for\_facial\_analysis.py**
2. **Code file:- CNN\_for\_real\_time\_facial\_analysis\_with\_alarm.py**

**3.2 Random Forest Model for Behavioural Analysis**

1. Created a **Random Forest model** trained on a dataset processed from synchronized video frames and eye-tracking data.
2. Extracted behavioural features such as:
   * **Eye Aspect Ratio (EAR)**
   * **PERCLOS (Percentage Eye Closure)**
   * **Head Position**
   * **pupil dilation and head roll/pitch variations**.
3. Trained the Random Forest Model and saved the model.
4. Code file:- random\_forest\_.py
5. Combined the **CNN model with the Random Forest model** to enable **real-time drowsiness detection** with an alert system.

**Implementation Details:**

1. **Data Loading & Feature Selection:**

* Loaded the merged drowsiness dataset from merged\_drowsiness\_data.csv using pandas.
* Selected relevant features for drowsiness prediction: EAR, PERCLOS, Blink rate, Pupil Diameter, Head Pitch, and Head Roll.
* Separated features (X) and target variable (y, "Label").

1. **Data Splitting & Cross-Validation:**

* Split the dataset into training and testing sets (80/20 split) using train\_test\_split.
  + Stratification was used to maintain the class distribution in both training and testing sets.
* Applied 5-fold cross-validation on the training data using cross\_val\_score to assess model generalization.
  + A Random Forest classifier with specific hyperparameters was used during cross-validation.
  + The mean cross-validation accuracy was calculated and printed.

1. **Model Training & Evaluation:**

* Trained a Random Forest classifier with the following hyperparameters:
  + n\_estimators=200: Increased the number of trees for better generalization.
  + max\_depth=10: Limited tree depth to prevent overfitting.
  + min\_samples\_split=10: Set the minimum samples required to split an internal node.
  + min\_samples\_leaf=5: Set the minimum samples required to be at a leaf node.
  + random\_state=42: Set a random seed for reproducibility.
* Evaluated the trained model on the test dataset using accuracy\_score.
* Printed the test accuracy.

1. **Model Saving & Real-Time Integration:**

* Saved the trained Random Forest model as drowsiness\_ml\_model.pkl using joblib.dump.
* **Real-time integration:**
  + The saved Random Forest model was integrated with a Convolutional Neural Network (CNN) for real-time drowsiness detection.
  + Real time testing was performed by streaming live data to the system, and processing this data through the CNN and Random Forest models.

1. **Code file:- CNN\_Random\_forest\_real\_time.py**

**3.3 Physiological Model Using LSTM**

* Collected physiological signals (**EEG, ECG, PPG, EDA**).
* Trained an **LSTM-based deep learning model** to analyse fatigue patterns.
* Code file:- F:\Project\Drowsiness Detection System\LSTM\_on\_Physiological.py
* Since real-time testing with actual sensors was unavailable, the model was evaluated using **randomly generated physiological data**.
* Code file:- LSTM\_real\_time\_on\_random\_data.py

**Implementation Details:**

1. **Data Loading & Merging:**

* Loaded EEG/ECG data from merged\_EEG\_ECG.csv and PPG/EDA data from merged\_PPG\_EDA.csv using pandas.
* Converted the 'time' column in both datasets to datetime objects for accurate time-based merging.
* Removed rows with missing 'time' values.
* Merged the two datasets using pd.merge\_asof with a nearest-match approach, synchronizing data based on the 'time' column.
* Removed the 'time' column, as it is not used as a model feature.
* Filled remaining missing values using forward-fill followed by backward-fill.

1. **Preprocessing & Normalization:**

* Separated the target variable ('Empatica/HR') from the feature set.
* Normalized the features using standardization (mean subtraction and division by standard deviation).
* Reshaped the feature data to the required LSTM input format (samples, timesteps, features), assuming a single timestep.
* Split the dataset into training and testing sets (80/20 split, random state 42).

1. **Model Training:**

* Defined an LSTM-based neural network model using TensorFlow/Keras.
  + Model architecture:
    - LSTM layer with 64 units, return sequences, and ReLU activation, followed by a Dropout layer (0.2).
    - LSTM layer with 32 units and ReLU activation, followed by a Dropout layer (0.2).
    - Dense output layer with 1 unit and linear activation.
* Compiled the model using the Adam optimizer (learning rate 0.001), mean squared error (MSE) loss, and mean absolute error (MAE) metric.
* Trained the model for 20 epochs with a batch size of 32, using the test set for validation.
* Saved the trained model as lstm\_model.h5.

1. **Evaluation & Real-Time Testing:**

* Evaluated the trained model on the test dataset, calculating and printing the MSE loss and MAE.
* Achieved [insert achieved test loss and mae here] on the test data.
* **Real-time testing:**
  + Simulated real-time data input by generating random data with the same feature structure as the training data.
  + Loaded the trained lstm\_model.h5 model.
  + Applied the same normalization used during training to the randomly generated data.
  + Used the model to predict HR values from the simulated real-time data.
  + This verified that the model can accept new data, perform normalization, and provide predictions.

1. **Code file:-** **LSTM\_real\_time\_on\_random\_data.py**

**3.4 NLP-Based Drowsiness Detection for Steering and Heart rate**

* Collected synthetic data to simulate heart rate and steering variability as indicators of drowsiness.
* Developed an **NLP-based machine learning model** to predict drowsiness based on heart rate and steering behaviour.
* Created a dataset of **100,000 synthetic samples** covering various heart rate and steering variability conditions.
* Trained and validated the model using this dataset, achieving high accuracy in drowsiness classification.
* Evaluated the NLP model on real-time generated inputs to simulate real-world scenarios.

**Implementation Details:**

1. **Synthetic Data Generation:**
   * Generated **100,000 samples** with **random heart rates (50-110 BPM)** and **steering variability (0.1-0.9)**.
   * Labelled data as **drowsy** if HR < 55 BPM or steering variability > 0.7.
   * Saved dataset as synthetic\_drowsiness\_data.csv.
2. **Preprocessing & Normalization:**
   * Loaded the dataset and extracted relevant features (Heartrate, Steering).
   * Used **StandardScaler** for normalization and saved the scaler for real-time use.
   * Split data into **80% training, 20% testing**.
3. **Model Training:**
   * Defined a **Neural Network** with:
     + **64 neurons (ReLU activation) + Dropout**
     + **32 neurons (ReLU activation) + Dropout**
     + **1 neuron (Sigmoid activation) for binary classification**
   * Compiled with **Adam optimizer & Binary Crossentropy loss**.
   * Trained for **50 epochs** with batch size **16**.
   * Saved the trained model as NLP\_model.h5.
4. **Evaluation & Real-Time Testing:**
   * Achieved **high accuracy** on test data.
   * The model is now capable of predicting drowsiness based on **heart rate and steering variability**.
5. **Code file:- NLP\_model\_real\_time\_stearing\_HR\_on\_synthrtic\_data.py**

**4. Model Integration & Real-Time System**

* Successfully developed four models:
  + **CNN Model (Visual Features)**
  + **Random Forest Model (Behavioural Features)**
  + **LSTM Model (Physiological Signals)**
  + **NLP Model (Speech Patterns)**
* Integrated all models into a **unified multimodal drowsiness detection system**.
* Enabled **real-time detection** with **minimal latency**.
* Implemented an **audio alert system** that beeps when drowsiness is detected.
* **Code file:**- final\_model.py

**5. Results & Future Improvements**

Successfully trained and tested **CNN, Random Forest, LSTM, and NLP models** for drowsiness detection. Developed a **real-time drowsiness detection system** with multimodal data fusion. Integrated an **audio alert system** for immediate driver warnings.

**Future Enhancements:**

* Integrate **real physiological sensors** (EEG, ECG, PPG) for real-world testing.
* Enhance **adaptive learning** for personalized drowsiness detection thresholds.
* Improve **speech-based NLP detection** for better fatigue recognition.

**6. Conclusion**

By combining **computer vision, machine learning, and physiological signal processing**, this multimodal system provides a **real-time, highly accurate, and adaptive drowsiness detection solution**. The integration of **visual, behavioural, physiological, and speech-based** features ensures robust performance, reducing road accidents and improving driver safety.

This project demonstrates the feasibility of real-time multimodal drowsiness detection and serves as a foundation for further enhancements using real-world physiological sensors and more sophisticated adaptive learning techniques.

