**Title: Real-time Machine Learning-Based Driver Monitoring and Safety System**

**ML Driver’s Shield or MLDS**

**Abstract:** This Project aims to create a Real time Machine Learning Based smart system that watches over drivers in real-time to keep them safe on the road by actively monitoring driver behaviour and taking preventive actions when necessary. Driver’s drowsiness and fatigue are the main reasons for most of the accident happening on the road. To overcome this problem and solve this, MLDS notifies the driver when signs of drowsiness are detected by a voice alert system and an audible alarm. MLDS uses Deep Learning technology, specifically CNN, for its operations. Even though cars have safety features like airbags, accidents still happen and sometimes it takes too long for help to arrive. The system can accurately detect the driver's eye status, monitor yawning patterns, and even respond by sending automatic notifications to the driver's relatives or friends if an airbag is deployed i.e., in case of accidents.

**Introduction:** The National Highway Traffic Safety Administration estimates that 100,000 accidents are the direct result of driver fatigue each year. 21 percent of all fatal accidents are due to drowsy driving. This Project aims to identify the driver’s drowsiness using Deep Learning techniques and alert the driver. This project uses a custom dataset built using images of my eyes and face. In this multifaceted project, it addresses three critical aspects of driver safety using a pretrained machine learning model renowned for its speed and accuracy, MobileNet, with the implementation of transfer learning. **Task 1: Driver's Eye Status Monitoring:** First task involves the real-time monitoring of the driver's eye status. By continuously analyse the driver's eyes. If the system detects closed eyes, it alerts the driver using an integrated alarm system. **Task 2: Airbag Status Monitoring:** a system to continuously monitor the status of the airbags. I have used a custom dataset comprising images of airbags deployed and non-deployed states for training. Employing CNN for image classification, to accurately determine whether the airbags have been deployed in case of accident. Then the system automatically sends alert messages to the driver's friends and relatives notifying them about the situation. **Task 3: Yawn Detection:** monitoring the driver's yawning behaviour as an indicator of fatigue. An innovative algorithm that identifies the driver's face and measures the distance between the upper and lower lips. When this distance surpasses a predefined threshold value, it signals a yawn and alerts the driver. What makes this project especially impressive is the System runs all three tasks at the same time in real-time, using just one camera to capture everything. It does this quickly and accurately. So, it always keeps an eye on the driver, the airbags, and yawning all at once to make sure everyone stays safe. By integrating these three vital components my project aims to enhance driver safety and provide peace of mind for both drivers and their loved ones

**Research Methodology**

In my project I have carefully researched and selected and developed specific methods, algorithms, architectures, and tools to tackle practical challenges, such as detecting eye status, identifying yawns, and recognizing airbag deployment from live camera feeds. These choices were made with a strong reason for getting good accuracy and real-time performance to ensure that my solutions are effective for real-world applications.

Below are the researched and selected algorithms, architectures, Library, and tools for this project.

1. **Classifier Selection**: Haar Cascade classifiers for the purpose of eye detection. The "eye Haar Cascade" classifier is trained to detect eyes in images or video frames in OpenCV. It has learned patterns and characteristics that are of eyes such as the arrangement of pixels, shapes, and contrast variations typically associated with eyes.

Choosing this as the main classifier because:

* Region of Interest (ROI): they can quickly identify ROI such as eyes in real time video frames.
* Real-time Performance: Haar Cascade classifiers are computationally efficient they provide real-time performance for applications requiring quick eye detection.
* Resource Efficiency: they typically require fewer computational resources compared to deep learning-based methods. This efficiency is advantageous for running the system on devices with limited processing power.

1. **Architecture Selection**: MobileNet architecture as a base model for binary classification, MobileNet is a deep convolutional neural network (CNN) primarily designed for efficient use on mobile and embedded devices while maintaining good performance for image classification tasks. MobileNet are small, low-latency, and low-power models that can be used for classification, detection, and other common tasks. MobileNet is an excellent choice for tasks like image classification, object detection, and feature extraction, especially for devices with limited computational resources.

Choosing this as a base model because:

* Low Latency and Real-time Performance: It is designed to be lightweight and computationally efficient. Its architecture includes depth-wise separable convolutions and point-wise convolutions which reduces the computational burden. As a result, MobileNet can provide real-time or near-real-time performance making it suitable for processing live camera feeds with low latency.
* Uses less computer memory or RAM: MobileNet use less memory which is great when you are working on devices without powerful GPUs. This efficiency helps the models run smoothly even on devices with limited computing power.
* High Accuracy in Compact Form: Despite its compact size and efficiency it still achieves a good accuracy in various image classification tasks, including eye status classification, airbag deployed or not, face wise recognition. Its depth-wise separable convolutions make it capable of providing good results for binary classification tasks.

1. **Library Selection for Lip Identification**: For the identification of lips to detect the yawns, I employ the dlib library which has facial landmark detection capabilities. Then use the dlib's shape predictor model that is trained to locate the 68 facial landmarks including the lips.

Choosing this library because:

* Accuracy and Precision: Dlib's shape predictor model is trained to locate 68 facial landmarks on face including the lips which offers high accuracy and precision in identifying facial features. Dlib's facial landmark detection is based on a combination of the Supervised Descent Method (SDM) and ensemble of regression trees. This allows to accurately locate facial landmarks including the lips in different poses and lighting conditions.
* Ease of Integration: ease of integration into Python-based computer vision pipelines. Its Python API simplifies the process of adding facial landmark detection into project.

1. **Tool Selection for Image and Video Processing**: OpenCV (Open-Source Computer Vision Library) is chosen as a fundamental tool for image and video processing in this Project because of its numerous advantages

Choosing the OpenCV library because:

* Frame Acquisition: To capture and read video frames from various sources, including webcams, video files, and streams
* Color Space Conversion: function for converting images and video frames between different color spaces, mostly grayscale images.
* Image Preprocessing: to perform various preprocessing tasks on frames, such as resizing, converting to grayscale, and Converting images to required extension. These steps enhance the quality of input data before analysis.
* Drawing and Visualization: to draw annotations and live reading on video frames, for displaying the results of predictions.

**Procedures**

1. Project Initiation
2. Literature review
3. Hardware and software setup
4. Data Collection Plan
5. Data Collection and Preparation
6. Data Cleaning and Preprocessing
7. Model Processing and Training
8. Analysing the Model's Performance
9. Testing the Model and Making Predictions on Unseen Data and on a Live Camera Feed
10. Individual Testing of Three Tasks
11. Integrating Alert Mechanism and Automatic Notification Mechanism
12. Testing and running All three task in a single framework
13. Integrating All Mechanisms and Running Predictions in a Single Framework
14. Exploration of Additional Work, Research, and Trial and Errors
15. Documentation
16. Reporting and Presentation