Driver Drowsiness Detection System

R. Bindu
Sreenidhi Institute of Science and Technology
Telangana, Hyderabad

M. Alekhya
Sreenidhi Institute of Science and Technology
Telangana, Hyderabad

Abdul Kareem
Sreenidhi Institute of Science and Technology
Telangana, Hyderabad

K. Karunakar
Sreenidhi Institute of Science and Technology
Telangana, Hyderabad

Abstract- **Driver drowsiness is a serious issue that poses a significant risk to road safety. To combat this problem, researchers and engineers have developed driver drowsiness detection systems that use advanced technologies to identify signs of drowsiness in drivers. Our proposed system is based on computer vision techniques and machine learning algorithms. The system is designed to analyze the driver's facial features in real-time, such as eye movements, head pose, and yawning, to detect signs of drowsiness. The system also includes an audio alarm that can alert the driver if drowsiness is detected. To develop and train our system, we used a large dataset of annotated driver drowsiness images. This dataset was used to train our machine learning algorithms to accurately identify signs of drowsiness in the driver's facial features. Our proposed system has several advantages over existing driver drowsiness detection systems. First, it is designed to work in real-time, which is essential for timely detection and intervention. Second, our system is highly accurate, thanks to the use of advanced computer vision techniques and machine learning algorithms. Finally, our system can be easily integrated into existing driver assistance systems, making it a practical and cost-effective solution. In conclusion, we believe that our proposed driver drowsiness detection system has the potential to significantly improve road safety by reducing the number of accidents caused by drowsy drivers. By providing an early warning system that can detect signs of drowsiness in real-time, we can help prevent accidents and save lives on our roads.**

Keywords: Convolutional Neural Network, Support Vector Machine (SVM), ReLU layers, Pooling layers, Keras, Dropout, Dense layer, Max pooling

I.INTRODUCTION

Transport networks are an integral part of modern life, enabling people to transport goods, get to work and discover new places. However, drowsiness or tiredness while driving is a serious problem that can have serious consequences, including accidents and death. In fact, drowsiness can affect a driver's alertness and reaction time, making them more prone to making mistakes and making wrong decisions behind the wheel. To solve this problem, it is important to design the with modern technology and develop systems that can closely monitor the driver's attention throughout the driving process. Such a system could be designed to track the open or closed condition of the driver's eyes and mouth. It is believed that by monitoring the eyes, early signs of drowsiness can be detected, allowing preventive measures to be taken to avoid accidents.
Deep learning algorithms such as convolutional neural networks (CNNs) could play a key role in the development of such systems. CNNs are a type of neural network that is particularly effective in computer vision because it can recognize patterns and identify features in images. They can also automate the process of extracting features, making them useful for analyzing large amounts of data. In CNN, the top layers correspond to the bottom layers of the pre-trained model, while the top layers are new layers that are tuned and trained to solve a specific problem. This allows the insights gained from the pre-trained model to be used as a starting point for a new model, enabling high-precision epoch learning. By using CNN to develop a driver alertness monitoring system, it is possible to create a reliable and effective tool to prevent accidents and ensure safe driving. This technology has the potential to significantly improve road safety and reduce the number of deaths and injuries caused by drowsiness or excessive fatigue at the wheel.

II. LITERATURE SURVEY

Driver drowsiness detection systems have been developed using various approaches, such as physiological signal analysis, eye movement analysis, and facial feature analysis. Among these, the facial feature analysis approach that uses computer vision and machine learning techniques is considered promising due to its high accuracy, real-time detection, and ease of integration into existing driver assistance systems.

III.EXISTING SYSTEM

Current drowsiness detection systems involve the use of devices that detect respiratory rate, heart rate, blood pressure, etc. These devices can cause discomfort to the driver while driving. We cannot be sure that drivers are constantly wearing all of these devices while driving. In this case, improper use of these systems can lead to poor accuracy of the final result. Your existing system may not work well in low light conditions. The driver's face and eyes may not be recognized, resulting in low accuracy. The existing system used a Support Vector Machine (SVM) to classify faces and determine whether a person was dozing or not. Several possible

hyperplanes can be chosen to separate the two classes of data points. Because the main goal of SVM is to find the plane that has the maximum margin, i.e., the maximum distance between the data points of the two classes. It goes straight to the driver's face and monitors the driver's eyes to detect fatigue. For large vehicles such as heavy trucks and buses, this setting is not

permanent. The bus has a large windshield to have a wide view for safe driving. This model also has low accuracy.

IV.PROPOSED SYSTEM

Evolving Neural Networks (CNNs) are deep learning models commonly used for image classification tasks. In this system, we use a CNN-based classification model to detect driver drowsiness. Unlike SVM, a machine learning algorithm, CNNs automatically learn and extract the appropriate features from the input data, eliminating the need for manual feature construction. However, CNNs require fixed-size images as input, and the video images captured by the camera may not be the same size. Then we need to pre-process the input by extracting keyframes from the video and storing them in the database. From these stored images, feature vectors are generated in convolutional CNN layers. These feature vectors represent the salient features of the input that are useful for detecting drowsiness. CNNs typically have multiple layers, including convolution layers, link layers, ReLU layers, and connected layers. In Convolutional Layers, the input data is convolved with a set of learnable filters to extract features from the input image. Pool levels are used to reduce the spatial size of the input through down sampling, which helps reduce the computation required in the network. ReLU (Rectified Linear Unit) layers introduce non-linearity into the network and help capture more complex features. Finally, the combined layers are used for classification by mapping the extracted objects to output classes.

In summary, the CNN-based classification model used in this system automatically learns and extracts relevant features from the input data, making it more accurate and efficient than traditional machine learning algorithms such as SVM. The pre-processing step of extracting keyframes from the video and generating feature vectors from the convolution layers ensures that the input data for the CNN model is correctly formatted.

V.SYSTEM ARCHITECTURE

The system architecture can be divided into four main stages: data acquisition, data preprocessing, feature extraction, and classification. The data acquisition stage involves capturing video data of the driver's face using a camera. The video data is then pre-processed in the data preprocessing stage to extract keyframes from the video and store them in the database. The keyframes are selected at regular intervals and represent the face images to be fed into the CNN-based classification model. In the feature extraction stage, the CNN model extracts the relevant features from the keyframe images. The CNN model consists of multiple convolutional layers, which convolve the input data with a set of learnable filters to extract features from the input image. Pooling layers are used to down sample the spatial size of the input, reducing the computation required in the network. ReLU layers introduce non-linearity into the network and help capture more complex features. In the classification stage, the feature vectors generated by the CNN model are used for classification. The combined layers are used to map the extracted features to output classes, such as alert or drowsy, based on the driver's state. The system can also include a user interface to display the real-time drowsiness level and provide feedback to the driver, such as suggestions to take a break or change driving behavior. In summary, the CNN-based classification model used in this system automatically learns and extracts relevant features

from the input data, making it more accurate and efficient than traditional machine learning algorithms. The pre-processing step of extracting keyframes from the video and generating feature vectors from the convolutional layers ensures that the input data for the CNN model is correctly formatted.

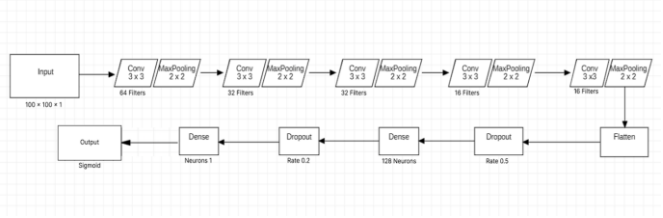

Fig: 1 CNN Architecture

Convolutional Neural Network:

Artificial neural networks (ANNs) are machine learning algorithms that are designed based on the neural networks present in the human brain. ANNs are able to learn complex patterns and relationships between classes of inputs and outputs. ANNs are becoming increasingly popular for image classification because they can extract useful features from images and use them to classify them into different categories. Convolutional Neural Networks (ConvNets/CNNs) are a type of ANN specifically designed for image recognition and classification tasks. They are made up of multiple layers, including a texture layer, a pool layer, and a fully connected layer. The pool layer is used to reduce the size of the feature maps generated by the convolution layer. This is accomplished by dividing the feature maps into smaller regions and taking the maximum or average value of each region. This reduces the computational load on the network and avoids overfitting. The fully connected plane is used to classify the input image into different categories. It takes flattened feature maps from the previous layer as input and applies different weights and distortions to get the final result. The CNN model used in this project is built using Keras, a popular deep learning library. The model is trained on a dataset of around 7000 images of eyes, which are labeled as either open or closed. During training, the model learns to extract useful features from the images and classify them into the correct category. In summary, CNNs are a powerful tool for image recognition and classification tasks, and the architecture of the network is designed to extract features from images in a hierarchical manner. With the help of pre-processing techniques and a well-designed CNN architecture, it is possible to build an accurate driver drowsiness detection system.

Dropout:
Attrition level is a regularization technique used in deep learning models to avoid overfitting. Overfitting occurs when the model is too complex and fits too tightly to the training data, resulting in poor generalization to new data. Dropout layering is a simple but effective technique that helps reduce overfitting by randomly turning off certain neurons during training. During training, the dropout level randomly selects a subset of neurons and discards them, meaning their outputs are set to zero. The rate parameter indicates the proportion of neurons to be released, with typical values between 0.1 to 0.5. During inference, all neurons are used and their output is reduced by a factor of 1/(1-rate) to ensure that the overall performance of the layer remains the same. The key idea behind the dropout layer is that it forces the network to learn more robust functions by preventing a single neuron from unduly affecting the output. By randomly eliminating neurons

during training, the network is forced to learn more independent representations of the input rather than relying on a few strongly correlated features. This makes the network more resilient to noise and allows for better generalization of new data. Layer dropout is a regularization technique that is widely used in deep learning and has proven successful in many applications including image classification, speech recognition, and natural language processing.

Dense layer:
A dense layer is a fully connected layer in which every neuron is connected to every neuron in the previous layer. It is the last layer of the neural network and helps generate the output. The dense layer receives the output of the previous layer and applies an activation function to the weighted sum of the inputs to produce the output. The main purpose of the dense layer is to combine the properties learned from previous layers and use them for predictions. This layer takes the high-level features extracted from the convolution layers and uses them to classify the input. The effectiveness of the dense layer depends on the number of neurons in the layer. Each neuron in the dense layer contributes to the final output by computing a weighted sum of the previous layer's outputs. The weights in this layer are learned during the training process and used to adjust the importance of each input in the final output. The activation function used in the dense layer is usually a nonlinear function such as ReLU, Sigmoid, or Tanh. This feature adds nonlinearity to the dense layer output, allowing the model to capture complex relationships between inputs and outputs. In summary, the dense layer is an important part of the neural network that helps generate the final result by combining the high-level features learned from previous layers. Its ability to capture nonlinear relationships in data makes it a powerful tool for classification tasks.

Pooling layers:
Pool layers are used to reduce the spatial dimensions (height and width) of feature maps created from convolution layers while preserving important information. This reduces the computational complexity of the model and avoids overfitting. In summarized layers, the input image is divided into non-overlapping regions or windows, and summary statistics are computed for each region. The most commonly used summary statistic is Max Pooling, which provides the most powerful functionality in a region. Another commonly used statistic is Average Pooling, which maintains the average value of a region. The size and spacing of the merge window can be adjusted to control the amount of spatial reduction in feature maps. Larger windows and seam steps result in a more aggressive spatial reduction, while smaller windows result in a more conservative reduction.

Max pooling:
Max pool is a type of pool level commonly used in CNN. The purpose of this layer is to reduce the spatial size of input feature maps by keeping only the most important information and discarding the rest. It works by dividing input feature maps into non-overlapping rectangular regions called pool areas or windows and, for each region in the pool, choosing the maximum feature map value in that region. For example, consider a pool level of 2x2 max. Given a 4x4 input feature map, the layer partitions the feature map into non-overlapping 2x2 pool regions. A maximum value is selected for each region and this value is placed at the appropriate location on the 2x2 output curve. The result of this process is a feature map that is half the size of the input feature map. Max pooling has several advantages. First, it reduces the spatial size of feature maps, which in turn reduces the number of parameters that need to be learned in subsequent layers of the network. Second, a degree of translation immutability is introduced, meaning that the output of the pool layer will be similar when the input feature map is translated slightly. Finally, it can also help reduce overfitting by introducing some degree of regularization.

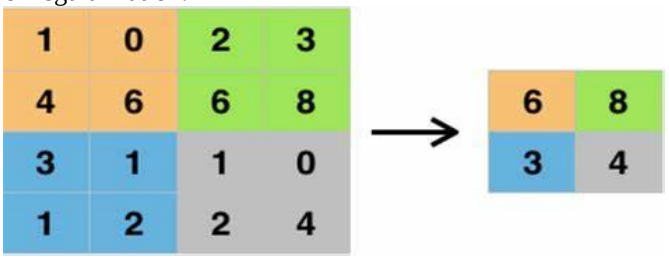

Fig. 2. Example Of Max pooling

ReLU layer:
The Haar Cascade detection algorithm is a machine learning approach used to detect objects in images or videos. Use a waterfall classifier trained for detection. As part of driver drowsiness detection, the algorithm may be used to detect the driver's face in the video stream from the camera facing the driver's face. Once a face is detected, it can be used as input to CNN's eye tracking and sleep tracking model. The Haar waterfall detection algorithm uses a series of positive and negative training examples to train the waterfall classifier. Positive samples are images that contain the object to be detected, while negative samples are images that do not contain the object. The algorithm uses a sliding window approach to scan the input image at different scales and positions, and applies a cascade classifier at each position to detect whether an object is present or not. When driver drowsiness is detected, the Haar cascade algorithm can be used to detect the driver's face in the video stream from the camera

facing the driver's face. Once a face is detected, it can be used as input to the CNN eye tracking and sleep tracking model.

Data Flow:
The below block diagram is the traditional Machine Learning Approach. It consists of two sections: the **training** and the **testing**. The training has the following components: the label, input, feature extractor, and the machine learning algorithm. The testing section has the following components in it: the input, feature extractor, the regression model, and the output label.

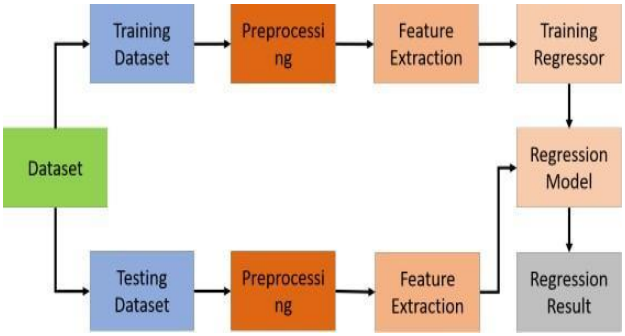

Fig 3: Block diagram for traditional machine learning approach

System flow:

1. First, we initialize the camera and capture the initial image.

2. We then recognize the face and eyes using the Haar Cascade algorithm.

3. The individual frames are then subjected to binarization. Binarization converts a multiscale image to a grayscale (black and white) image.
4. We then apply the Circle Hough transform to these frames to detect the pupils. Circle Hough Transformation is a basic feature extraction technique, used to detect circles in improper images.

5. Once the pupils are detected, we set a threshold value for continuous pupil visibility.

6. If the pupil is not detected for a period of time that is greater than the threshold, then drowsiness is detected and the alarm system is triggered.

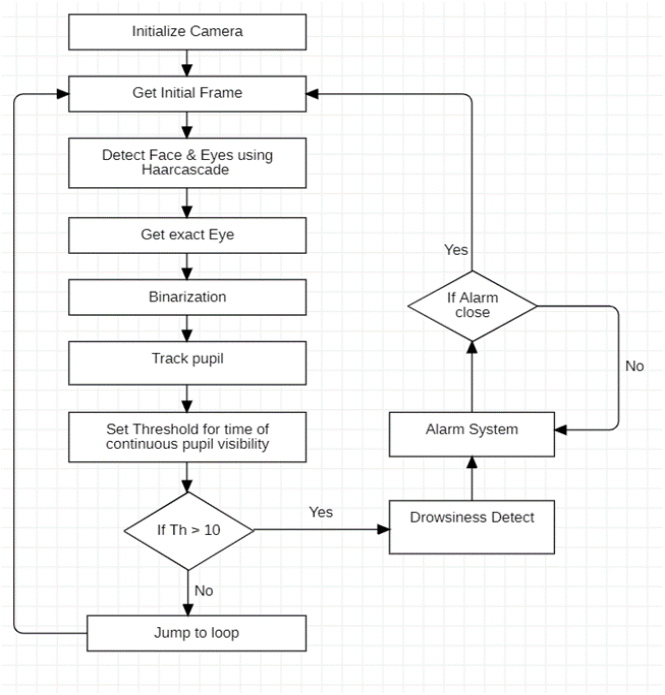

Fig 4: Flowchart of the system

V.TRAINING THE MODEL

Machine learning algorithms need training data to learn how to classify or predict the output. This is where learning models come into play. A training model is a set of data used to train an algorithm to recognize patterns in input data and produce accurate output. The training model contains input and output data. The input data is used to train the algorithm, while the output data provides correct answers or expected results for comparison. The algorithm processes the input data to produce the output data, and the output data is compared to the expected results to determine the accuracy of the model. The accuracy of the training dataset or validation dataset is critical to the accuracy of the model. In supervised learning, the learning data includes both inputs and outputs, and each data set with an expected input and output is called a supervised signal. The training is based on the discrepancy between the processed result and the documented result when feeding the input data into the model. The model uses input-output pairs to determine how to predict new, never-before-seen data. The aim is to minimize the gap between expected production and actual production. On the other hand, unsupervised learning does not require labeled output. Instead, the algorithm learns to identify patterns or clusters in the input data.

The algorithm processes the input data to discover hidden structures, relationships, or patterns in the data. Similar data is then grouped into clusters. The aim is to discover new patterns or structures in previously unknown data. Unsupervised learning is useful in cases where the outcome is not easily measurable or accessible, or when the data is too complex for expert analysis.

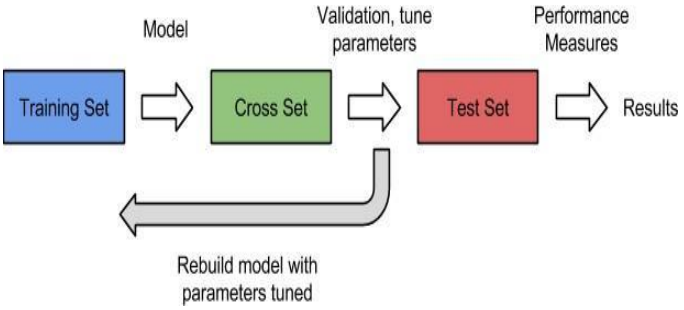

Fig 5: Training the model

VI. Methods / Algorithms

Haar Cascade Algorithm:

Haar cascades are widely used for face detection in computer vision and image processing applications. The algorithm was proposed by Viola and Jones in their seminal paper in 2001. The algorithm is trained on a large dataset of positive images consisting of faces and a large dataset of negative images not containing any face. The training dataset is used to learn features that are useful for detecting faces in new images or video streams. The Haar cascade algorithm works by scanning an image with a sliding window of various sizes and aspect ratios. At each position of the sliding window, a set of features is extracted from the image. These features are simple rectangular areas of the image with specific intensity properties, such as edges or contrast changes. The algorithm then evaluates these features using a cascade of classifiers. Each classifier is a machine learning model that uses a set of weights to evaluate the features and determine if there is a face present in the image. If a classifier determines that a face is not present, the algorithm moves on to the next position of the sliding window. If a classifier determines that a face may be present, the algorithm continues to evaluate the features with the next classifier in the cascade. The cascade is designed to be efficient and quickly eliminate non-face regions of the image. The Haar cascade algorithm is fast and efficient, making it suitable for real-time face detection applications. The algorithm is also capable of detecting faces at different scales and orientations. However, the algorithm is not perfect and may generate false positives or fail to detect faces in some images. Therefore, it is important to use the algorithm in conjunction with other methods for reliable face detection.

In summary, Haar cascades are a powerful technique for face detection in images and video streams. The algorithm is based on simple rectangular features that are efficiently evaluated using a cascade of classifiers. The algorithm is widely used in computer vision and image processing applications and has paved the way for many other object detection algorithms.

Making a Haar Cascade Classifier

The algorithm can be explained in four stages:

- Calculating Haar Features
- Creating Integral Images
- Using Adaboost

- Implementing Cascading Classifiers

Note that this algorithm requires multiple positive face images and negative faceless images to train the classifier, like other machine learning models.

Calculating Haar Features:

Haar Falls is used to detect objects in real-time images or videos. The algorithm uses Haar properties to detect patterns in the input data. The Haar function is essentially a calculation performed on rectangular areas adjacent to a specific position within the acquisition window. The calculation consists of adding the intensities of the pixels in each area and calculating the differences between the sums. These properties can be difficult to determine in the case of a large image. This is where integral images come in, as the number of operations is reduced by using an integral image. The next step is to train the hair waterfall classifier. This is to provide the algorithm with a large number of positive and negative images. Positive images are those that contain the detected object, while negative images do not contain the object. The algorithm uses positive images to learn more about an object's patterns and properties, while negative images help it understand what an object is not. During training, the algorithm creates a series of classifiers, each designed to recognize a specific aspect of the detected object. Each classifier consists of many weak classifiers, each using a specific Haar characteristic to recognize an object. These classifiers are combined in a waterfall structure to create a powerful classifier that can accurately recognize an object. Once the classifier is trained, it can be used to detect an object in new images or videos in real time. The image or video is first processed at multiple scales to detect objects of different sizes. The hair-waterfall classifier is then applied to each point in the image or video, using several weak classifiers to determine whether an object is present. When an object is detected, a bounding box is drawn around it to highlight its position.

In summary, the Haar-Waterfall algorithm involves collecting Haar features, training a waterfall classifier using positive and negative images, and then using the classifier to recognize an object in new images or videos. The algorithm is widely used in computer vision and image processing applications due to its speed and accuracy.

Creating Integral Images:

Integral images are an important concept in the Haar waterfall algorithm because they greatly reduce the computational time and power required to compute Haar features. An integral image is a data structure created from the original image by performing a simple operation on each pixel value. This operation calculates the sum of all pixel values above and to the left of the current pixel. This sum is then stored in the corresponding pixel of the integral image. After the integral image has been constructed, the Haar functions can be computed much more efficiently. The Haar feature is essentially a rectangular area in the image, and the feature value is calculated by subtracting the sum of the pixel values in the white rectangle from the sum of the pixel values in the black rectangle. This calculation can be performed in constant time using an integral image since the sum of the pixel values in each rectangular area can be calculated using only four references to the integral image. The use of integral images significantly

speeds up the calculation of Haar characteristics and enables efficient detection of objects in the images. Instead of computing Haar features at every possible position and size in the image, the algorithm can efficiently evaluate a small subset of candidate regions using computed integral images. This significantly reduces the algorithm's computational requirements, making it practical for real-time applications such as face detection in video streams.

Adaboost Training:
Adaboost or Adaptive Boosting is a machine learning algorithm used to create a strong classifier by combining multiple weak classifiers. In the context of object detection, Adaboost selects the best features of hair and trains weak classifiers to use them. To do this, it iteratively selects the features that provide the most information and uses them to train the classifiers. The algorithm begins by initializing the weights of each sample in the training set. Then choose a weak classifier based on the weighted training data. This weak classifier attempts to classify data into positive and negative samples using a threshold. If the classifier succeeds in classifying the data, its weight is increased and the weight of misclassified samples is decreased. The process continues by iteratively updating the weights and choosing a new weak classifier until a certain number of iterations is reached or the classification error reaches an acceptable level. Weak classifiers are combined into a strong classifier using cascading classifiers arranged one after the other. The waterfall consists of a series of steps, each containing a set of weak classifiers. The output of each step is used to decide whether to proceed to the next step or not. If an object is detected in the first pass, the algorithm does not proceed to the next pass. This leads to faster object detection since the algorithm does not have to calculate all features for each window in the image. In summary, Adaboost is a powerful machine learning algorithm used to build powerful classifiers for object detection. It does this by selecting the best features and training weak classifiers to use them, which are then combined into a strong classifier using cascading classifiers. The use of Haar characteristics and integral images speeds up the computation of the algorithm, making it suitable for real-time object detection.

Implementing Cascading Classifiers:
Haar cascades use cascading classifiers to efficiently detect objects in an image or video. The cascade classifier is a series of stages, each containing a collection of weak learners. The weak learners are trained using boosting, which allows for a highly accurate classifier by combining the mean prediction of all weak learners. The first stage of the cascade classifier processes the image with the most discriminative features. If an image fails this stage, it is rejected and skipped to the next stage, which processes the image with less discriminative features. This process continues until all stages are processed or an object is detected. The primary objective of the cascade classifier is to quickly reject regions of the image that do not contain the object of interest. This is because the majority of

regions in an image will not contain the object being searched for. The stages of the cascade are designed to reject negative samples as quickly as possible, while maximizing the true positive rate. The output of the cascade classifier is a set of bounding boxes around the detected object. These bounding boxes can then be used to track the object or perform further analysis. Haar cascades have many practical applications, such as face detection in images or videos. They are fast and accurate, making them useful in real-time applications. However, they do have limitations, such as difficulty in detecting objects with complex shapes or when the object is concluded. Overall, Haar cascades and the cascade classifier algorithm have made significant contributions to the field of computer vision and image processing, allowing for efficient and accurate object detection in various applications.

VII. CONCLUSION

Based on the project on training machine learning models to detect driver drowsiness using image processing, the following conclusions can be drawn:

- The data must be cleaned and organized before training the machine learning models to ensure that all features are in a suitable format.
- The developed system is non-invasive and uses an image processing algorithm to locate the driver's eyes and monitor fatigue.
- The system is capable of determining if the driver's eyes are open or closed during monitoring, and a warning signal is sent if the eyes remain closed for an extended period.
- Image processing is an accurate and reliable method for detecting driver drowsiness, and it offers a non-intrusive approach to monitoring driver alertness based on continuous eye closures.

In summary, this project has demonstrated the potential of image processing in detecting driver drowsiness and improving road safety. The developed system provides a reliable and non-intrusive solution for monitoring driver alertness, and it can be further enhanced with additional modifications and improvements.

VIII. FUTURESCOPE

The current state of this technology indicates that it is still in its early stages of research and development. However, there have been some initial findings that suggest certain modifications could be made to improve its effectiveness. Capture individual drivers steering activity while drowsy. The current system for detecting driver drowsiness using image processing has potential for further development. One possible extension is the addition of security measures, such as restricting access to the vehicle to only authorized individuals. In the event of theft, the system could prevent the vehicle from starting and also send an MMS containing a picture of the thief to the vehicle owner as a form of alert. This could enhance the security and safety aspects of the system. Perform more simulation experiments to verify the algorithm, evaluate it under different road conditions, and test it on a wider range of drivers. The above content suggests conducting further simulator experiments to validate the algorithm, which includes testing it under different road conditions and a more diverse group of drivers.

IX. REFERENCES

[1] Driver_Drowsiness_Recognition_via_3D_Conditional_GAN_and_Two-Level_Attention_Bi-LSTM https://ieeexplore.ieee.org/document/8926344

[2] CNN concept and architecture **https://www.youtube.com/watch?v=E5Z7FQp7AQQ**

[3] Drowsiness detection system **https://www.youtube.com/watch?v=E5Z7FQp7AQQ**

[4] **https://ieeexplore.ieee.org/search/searchresult.jsp?action=search&newsearch=true&matchBoolean=true&queryText=(%22All%20Metadata%22:drowsiness%20detection)%20AND%20(%22All%20Metadata%22:machine%20learning)%20AND%20(%22All%20Metadata%22:transactions)&ranges=2020_2023_Year**

[5] Assari, M. A., & Rahmati, M. (2011). Driver drowsiness detection using face expression recognition. 2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA).

[6] Dwivedi, K., Biswaranjan K & Sethi, A. (2014). Drowsy driver detection using representation learning. 2014 IEEE International Advance Computing Conference (IACC).

[7] Subbarao, A., Sahithya, K. (2019) Driver Drowsiness Detection System for Vehicle Safety, International Journal of Innovative Technology and Exploring Engineering (IJITEE)

[8] Sukrit Mehta, Sharad Dadhich, Sahil Gumber, Arpita Jadhav Bhatt (2019). Real-Time Driver Drowsiness Detection System Using Eye Aspect Ratio and Eye Closure Ratio International Conference on Sustainable Computing in Science, Technology and Management

[9] Rajasekar.R, Vivek Bharat Pattni, S.Vanangamudi "Drowsy driver sleeping device and driver alert system", IJSR, Vol.3 Issue4,2014

[10] Ramalatha Marimuthu, A. Suresh, M. Alamelu and S.Kanagaraj "Driver fatigue detection using image processing and accident prevention", International journal of pure and applied mathematics, vol. 116,2017