**A PROJECT ON**

****

**DRIVER DISTRACTION DETECTION SYSTEM BASED ON SVM AND PCA WITH PRIORITIZED ALERTING**

****

**Submitted in partial fulfillment of the requirement for the award of the degree of**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & ENGINEERING**

**Submitted by:**

**Ruudra Amola** **2016972**

**Shailja Bahuguna** **2017014**

***Under the Guidance of***

**Mr. Rishi Kumar**

**Assistant Professor, Department of CSE**

**Project Team ID: MP23CSE053**



**Department of Computer Science and Engineering**

**Graphic Era (Deemed to be University)**

**Dehradun, Uttarakhand**

**June-2024**



**CANDIDATE’S DECLARATION**

We hereby certify that the work is being presented in the Project Report entitled **“Driver distraction detection system based on SVM and PCA with prioritized alerting”** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering and submitted to the Department of Computer Science and Engineering of the Graphic Era (Deemed to be University), Dehradun is an authentic record of our work carried out during a period from **August-2023 to May-2024** under the supervision of **Mr. Rishi Kumar**, **Assistant Professor**, Department of Computer Science and Engineering, Graphic Era (Deemed to be University).

The matter presented in this dissertation has not been submitted by us for the award of any other degree of this or any other University.

Ruudra Amola 2016972

Shailja Bahuguna 2017014

This is to certify that the above statement made by the candidate is correct to the best of our knowledge.

**Supervisor** **Head of the Department**

**External Viva**

**Name of the Examiners: Signature with Date**

1.

2.

**Abstract**

The road accident scenario in India can be understood by simple statistics. and that every year more than 1.5 Lakh people lose their lives on the road. This is like crashing a jumbo jet carrying 340 people every day without any survivors. This is the enormity of the situation. Road accidents impact India’s GDP by 3.14% (IIT Delhi- TRIPP Study).

This paper mainly focuses on the solution to the issue of distracted driving. Distracted driving is an act of indulging in attention-breaking activities while driving that can disrupt the flow of the driver’s coordination on the road. These distractions can be broadly classified as Manual, Visual, and Cognitive. The following research work shows how driver distractions can be detected using computer vision equipped cameras in the vehicle to detect an act of distraction. We have used the State Farm Distracted Driver Detection Dataset (SFDDD), which consists of a set of images classified into 9 categories of driver distraction activities. We have adopted a Machine Learning approach and have trained a classification model using Linear Support Vector Machine (SVM), achieving an accuracy of 98.95%. The result has a trade-off with the time and space and was further improved by merging SVM with Principal Component Analysis (PCA) that significantly reduced the model training time and even enhanced the accuracy to 99%. Using the model, we have also implemented a prototype for prioritized alerting system that warns the driver as soon as it detects a distraction and sets off an alarm.

After a thorough research and analysis of traffic-related statistics, we have also proposed an extension to the dataset by including a list of 8 more forms of driver distractions which can significantly broaden the classification and help in improving the accuracy of a detection model created using the State Farm Distracted Driver Detection Dataset.

**Keywords:** fatality, distracted driving, manual, visual, cognitive, machine learning, SFDDD, SVM, time, space, PCA.

**Acknowledgement**

Any achievement, be it scholastic or otherwise does not depend solely on individual effort but on the guidance, encouragement, and cooperation of intellectuals, elders, and friends. Several personalities in their capacity have helped me in carrying out this project work.

Our sincere thanks to project guide **Mr. Rishi Kumar, Assistant Professor,** Department of Computer Science and Engineering, Graphic Era (Deemed to be University), for his valuable guidance and support throughout the course of project work and for being a constant source of inspiration.

We extend our thanks to **Prof. (Dr.) Guru Prasad M.S.**, Project Coordinator, Department of Computer Science and Engineering, Graphic Era (Deemed to be University), for his valuable suggestions throughout all the phases of the Project Work.

We are extremely grateful to **Prof. (Dr.) D. P. Singh**, HOD of the Computer Science and Engineering Department, Graphic Era (Deemed to be University), for his moral support and encouragement.

We thank the **management of Graphic Era (Deemed to be University)** for the support throughout the course of our bachelor’s degree and for all the facilities they have provided.

Last, but certainly not least we thank all teaching and non-teaching staff of Graphic Era (Deemed to be University) for guiding us on the right path. Most importantly we wish to thank our parents for their support and encouragement.

Ruudra Amola 2016972

Shailja Bahuguna 2017014

**Table of Contents**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | | **Contents** | **PageNo.** | | Abstract | i | | Acknowledgement | ii | | Table of Contents | iii | | List of Tables | v | | List of Figures | vi | | **Chapter 1 Introduction** | **1-4** | | 1.1 Project Introduction | 1 | | 1.2 Problem Statement | 3 | | 1.3 Objectives | 4 | | **Chapter 2**  **Literature Survey/ Background** | **5-8** | | **Chapter 3 Software Design** | **9-14** | | 3.1 UML Diagrams of Proposed Model | 9 | | 3.1.1 Class Diagram | 9 | | 3.1.2 Object Diagram | 10 | | 3.1.3 Use-case Diagram | 11 | | 3.2 Control Flow Diagram of Proposed Model | 13 | | **Chapter 4**  **Requirements and Methodology** | **15-24** | | 4.1 Requirements | 15 | | 4.1.1 Hardware Requirements | 15 | | 4.1.2 Software Requirements | 15 | | 4.2 Methodology | 16 | | 4.2.1 Dataset and proposed classes for dataset extension | 16 | | 4.2.2 Dataset Visualization and Preprocessing | 18 | | 4.2.3 Dimensionality reduction using PCA | 19 | | 4.2.4 Support Vector Machine (SVM) Algorithm | 20 | | 4.2.5 Model Training and Approach | 21 | | 4.2.6 Implementation of Auditory alerting System | 22 | | **Chapter 5**  **Coding /Code Templates** | **25-25** | | 5.1 Directory Structure | 25 | | 5.2 Directory Descriptions | 26 | | 5.2.1 `/App` directory | 26 | | 5.2.2 `/imgs` directory | 30 | | 5.2.3 `/static` directory | 31 | | 5.2.4 `/templates` directory | 31 | | 5.2.5 `/TrainingPhase` directory | 32 | | **Chapter 6**  **Testing** | **33-37** | | 6.1 Model evaluation library: scikit-learn | 33 | | 6.2 Manual Testing | 34 | | 6.2.1 Normal Test Cases | 34 | | 6.2.2 Edge Test Cases | 36 | | **Chapter 7** **Results and Discussion** | **38-42** | | 7.1 User Interface and Output Screens | 38 | | 7.1.1 Home Page | 38 | | 7.1.2 Main Page | 39 | | **7**.1.3 Prediction Output | 40 | | 7.2 Model Evaluation and Discussion | 41 | | **7**.2.1 Confusion Matrix | 41 | | 7.2.2 Comparative Analysis of Performance: PCA vs. Non-PCA-based Approaches | 41 | | **Chapter 8 Conclusion and Future Work** | **43-44** | | 8.1 Conclusion | 43 | | 8.2 Future Work | 44 | | **Details of Research Publication** | **45** | | **References** | **46-48** | |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **List of Tables**   |  |  |  | | --- | --- | --- | | **TABLE No.** | **TITLE** | **PAGE No.** | | 2.1 | Table of related work | 6 | | 4.1 | Driver Distraction Categories from Dataset | 16 | | 4.2 | Additional Driver Distraction Classes | 17 | | 4.3 | Hierarchy scale of fatality levels | 23 | | 7.1 | Comparative Analysis of SVM performance | 42 |   **List of Figures**   |  |  |  | | --- | --- | --- | | **FIGURE No.** | **TITLE** | **PAGE No.** | | 3.1 | Class Diagram of Flask Web Application | 10 | | 3.2 | Object Diagram of Flask Web Application | 11 | | 3.3 | Use-Case Diagram of Distracted Driver Detection System | 12 | | 3.4 | Control Flow Diagram | 13 | | 4.1 | Bar Plot Sampling from Class Distribution | 18 | | 4.2 | Number of Principal Components and their cumulative variance | 20 | | 4.3 | Linearly and Non-Linearly Separated Spaces | 20 | | 4.4 | System Diagram | 22 | | 4.5 | UI of the System | 23 | | 5.1 | Directory Structure | 25 | | 5.2 | Code Snippet of instances ‘scalar’ and ‘pca’ | 26 | | 5.3 | Code Snippet of trained model | 26 | | 5.4 | Code Snippet of ImageProcessor class | 27 | | 5.5 | Code Snippet of PredictionModel class | 27 | | 5.6 | Code Snippet of AudioAlert class | 28 | | 5.7 | Code Snippet of DistractionDetector class | 28 | | 5.8 | Code Snippet of detect\_distraction() function | 29 | | 5.9 | Code Snippet of get\_distraction\_category() function | 29 | | 5.10 | Code Snippet of handle\_alert() function | 29 | | 5.11 | Code Snippet of Prediction Route (/predict) | 30 | | 5.12 | Code Snippet of form in index.html | 31 | | 6.1 | Comparison of original and predicted labels | 33 | | 6.2 | Normal Test Case 1 (Safe Driving) | 34 | | 6.3 | Fatality Level Identified and Alerting Latency | 34 | | 6.4 | Normal Test Case 2 (Reaching Behind) | 35 | | 6.5 | Normal Test Case 3 (Talking on the phone – left) | 35 | | 6.6 | Fatality Level Identified and Alerting Latency | 35 | | 6.7 | Normal Test Case 4 (Operating the radio) | 36 | | 6.8 | Fatality Level Identified and Alerting Latency | 36 | | 6.9 | Edge Case 1 (Invalid file) | 36 | | 6.10 | Edge Case 2 (Nothing Uploaded) | 37 | | 7.1 | Home page Screenshot | 38 | | 7.2 | Main page Screenshot | 39 | | 7.3 | File Dialog | 39 | | 7.4 | Prediction Output | 40 | | 7.5 | Fatality Level Identified and Alerting Latency | 40 | | 7.6 | Confusion Matrix for SVM with PCA | 41 | | A | Proof of Communication ICCUBEA | 45 | |  |

**Chapter 1**

**Introduction**

* 1. **Project Introduction**

Driving is an act of controlling or operating a vehicle on land. It involves controlling the speed, movement, and direction of a vehicle. Driving is done by a person, known as a driver, manually using equipment attached to the vehicle. Driving is an extremely sensitive activity that involves a strong level of focus and coordination of motor and sensory organs. The driver needs to maintain undivided attention and must be calm and tranquil in nature while driving on the road. Any kind of distraction, for instance, if the driver takes his eyes off the road while driving, even for a fraction of a second, can have serious ramifications [1][2].

Distracted driving is an act of indulging in attention- breaking activities while driving that can disrupt the flow of the driver’s mental and physical coordination and can lead to difficulty in driving and maintaining focus on the road. It creates a perilous situation, not only for the driver but also for fellow passengers, pedestrians, and people driving other vehicles on the road[3][4].

Recent years have seen a significant rise in the number of vehicles running on the roads and the growing use of technology and devices like mobile phones have contributed much to the issue of distracted driving. Researchers and experts from around the world have expressed keen interest in solving this problem and have proposed their solutions and ideas to tackle the issue[5].

Studies have concluded that the sources of distractions may lie inside the vehicle or outside of it. It can be equipment-related or otherwise and may be self-encouraged or imposed on the driver under certain circumstances. Sources of distractions can be widely classified under three categories: Manual, Visual, and Cognitive [22].

Manual distractions are the ones where the driver may have eyes on the road but is doing some manual activity while driving like taking his hands off the steering wheels, searching for something in the dashboard, attempting to reach the backseat of the car, and more[1][6].

Visual distractions, as the name suggests, relate to visual distractions in and around the vehicle like the driver trying to repeatedly look at the GPS navigator, looking at roadside billboards, and other surroundings, and looking at the rearview mirror for too long.

Cognitive distractions are the most fatal ones as they cannot be seen or detected, they occur in a person's mind. It can be referred to as mental absence while driving, thinking of something that disturbs the driver’s focus[7].

According to the records laid by the World Health Organization, road accidents are the 9th leading cause of death. India is a country where the frequency of accidents is unmatchable to any other nation, an accident happens every minute and a life is lost every four minutes. One of the most prominent causes of driver distraction is the use of mobile phones, whether it is for calling or for GPS navigation. A previous study revealed that 31% of the drivers who used a mobile phone while driving met with an accident. The current study analyzed and modeled the effects of mobile phone distraction on the reaction time of drivers belonging to different age groups in India[8][9].

A research study performed by the Indian Institute of Technology Bombay (IITB) has shown how using technical equipment like mobile phones while driving can distract drivers and affect their ability to handle fatal situations on the road by reducing their reaction times. The results of the study have shown that both calling and texting while driving degrades the ability to react to a driver [6].

In a significant advancement to embedded distraction detection systems, German technology firm Bosch developed what it describes as an “interior monitoring system” for cars that utilize the extensive computer vision approach that is based on the idea of merging artificial intelligence (AI) and cameras. One of its features included a camera integrated into a steering wheel. This can detect when a driver is distracted, when their eyes are becoming heavy that is the situation of drowsiness, and when they are trying to reach out to the rear seats or the person sitting next to them [7][10].

But the solution is yet to touch the wider mass. Firstly, the problem of distraction is still not clear to drivers on the road. It is still casual for people to have snacks and drinks while driving, listen to loud music, and text while driving without realizing the seriousness of the events it can lead up to. Secondly, all these brands have implemented the systems with mostly one feature which is fatigue, drowsiness of the driver, and eye-head movements. But the reasons for distraction are not limited to drowsiness only, there is much more to counter.

Distracted driver classification primarily involves two main approaches. The first approach involves using wearable setups with sensors that measure brain signals, heartbeat, and voluntary actions. However, this approach proves to be cost- effective and uses complex hardware. The other idea involves using camera vision systems to detect and predict the type of distraction. It commonly uses deep learning algorithms to perform feature extraction and classification tasks. While the first approach can detect cognitive distractions, the second one can detect manual and visual distractions. In this research work, we try to explore the second approach in a much more open fashion.

State Farm is an insurance firm in the United States. They surveyed with the help of some subjects, both male and female and created a dataset of images depicting different types of distracting activities that drivers tend to indulge in. State Farm aimed to improve these statistics, and better ensure their customers, by testing how dashboard-fixed cameras can automatically detect drivers' distracted behaviors.

In April 2016, State Farm initiated a competition on the website called Kaggle to collect ideas and solutions to the driver’s behavior problems using their dataset. We have used the same State Farm driver dataset to train our proposed machine learning model.

Our problem focuses on detecting distracted acts of car drivers based on the State Farm driver distraction image dataset from Kaggle, using Machine Learning techniques to achieve maximum accuracy. The dataset is widely classified into 10 classes that represent 9 different causes of hindrances in driver attention. The 10th class represents “safe driving”.

* 1. **Problem Statement**

The **problem statement** for the present work can be stated as follows:

The project addresses the critical issue of detecting distracted driving behaviors using the State Farm Driver Distraction Image Dataset from Kaggle. The dataset comprises images classified into 10 categories, representing nine different types of driver distractions (e.g., texting, talking on the phone, eating) and one category for "safe driving".

The primary goal of this project is to develop and optimize Machine Learning (ML) models to accurately classify these images into their respective categories and achieving maximum accuracy in image classification which is crucial for the effectiveness of the model. Additionally, the project aims to implement an auditory alert system that notifies drivers based on the severity of the detected distraction, thereby mitigating potential accidents.

By leveraging existing ML algorithms, the project aims to create a strong system capable of real-time detection and alerting, ultimately contributing to safer driving practices.

* 1. **Objectives**

The potential work objectives are as follows:

1. To perform comprehensive visualization and initial analytics of the collected image dataset. This step aims to understand the distribution and characteristics of the data, which will help in subsequent model development.
2. To develop a robust classification model using Linear Support Vector Machine (SVM). The goal is to accurately classify images into the 10 predefined categories.
3. To optimize the performance and accuracy of the Linear SVM model by incorporating Principal Component Analysis (PCA) over the vector components, which is a dimensionality reduction technique.
4. To establish a hierarchy of distraction priority levels based on the potential fatality risk associated with each type of distraction. For example, texting while driving may be classified as high-priority, while talking on the phone might be considered lower-priority. This hierarchy will help the creation of an auditory alert system.
5. To design and implement a prototype auditory alert system that provides real-time warnings to drivers based on the severity of the detected distraction. This involves developing a sample user interface (UI) that integrates with the alert system.
6. To finely evaluate the performance of the developed models using appropriate metrics. This objective includes validating the models against a test dataset to ensure their reliability and effectiveness in the real-world.

**Chapter 2**

**Literature Survey/ Background**

**This section outlines the research work on distracted driver detection in the last few years. Authors in [10] presented a review paper on Driver Drowsiness Detection Systems and identified some measures for the detection of distracted drivers. “Biological-based measures” involve the use of wearable sensors to detect the driver’s drowsiness using brain activity, heart rate, and muscular activities. While wearable technology is expensive and requires user involvement, “Image-Based Measures” utilizes a camera for vision-based classification. Authors in [11] employed SVM and logistic regression models to develop a real-time approach for detecting driver distraction. The SVM model outperformed traditional logistic regression models, achieving an average accuracy of 81.1%. In the research work proposed by authors in [27] they performed feature extraction through CNN and used PCA before the training of SVM model achieving an accuracy of 96.28%. Authors in [12] introduced a deep learning-based CNN model that collected the images for the dataset using a Kinect camera. The study utilized self-made feature extractors integrated with classifiers like SVM and CNN.**

**However, CNN-based continuously outperformed SVM. The superiority of CNN was attributed to its enhanced efficiency and faster computational speed. The problem has also been studied using various Deep- learning approaches to achieve better results. Deep learning approaches involved models such as AlexNet, and convolutional neural network (CNN) architectures like ResNet50, VGG-16, VGG-19, and GA-weighted ensemble. Individual CNN models tend to improve accuracy over a better scale on raw image datasets. Authors in [13] proposed the use of the Res-Net 50 network to classify the images of distracted drivers along with a tag that represents the type of distraction in a video. Their approach achieved an accuracy of 94%.**

**In 2023, research [14] which focused on detecting driver behavior with a combination of artificial deep learning and machine learning models with GA (genetic algorithm), achieved an accuracy of 99.80%. Overall, when compared to conventional machine learning approaches, deep learning models outperform them. Deep learning can help to identify a driver’s stress levels and behavioral patterns by tracing eye movements and physical movements.**

**Some of the significant research works that have led to a paradigm shift in the study of driver distraction detection systems have been summarized in Table 2.1.**

**Table 2.1 Table of related work**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Year** | **Author** | **Title** | **Contribution** | **Algorithm** | **Advantage** | **Weakness** |
| 1 | 2018 | Prof. Pramila M. Chawan, Shreyas Satardekar, Dharmin Shah, Rohit Badugu, Abhishek Pawar | Distracted Driver Detection and Classification | Use of more than one model to decrease the log loss value. | An ensemble was created after averaging the probabilities generated by VGG-16, VGG-19 and InceptionV3. | The final log loss value was  0.795. | Accuracy of model needs improvement. |
| 2. | 2021 | Alexey  Kashevnik, Roman Shchedrin, Christian Kaiser, Alexander Stocker | Driver Distraction Detection Methods: A Literature Review and Framework | Reviewed driver distraction approaches. | Provided an overview of both driver distraction research and driver distraction detection systems. | Detected 3 main classes of distraction-manual, visual and cognitive. | More study can be done on this. |
| 3. | 2021 | Faiqa Sajid, [Abdul Rehman Javed](https://ieeexplore.ieee.org/author/37088389492), Asma Basharat, Natalia Kryvinska, Adil Afzal,  Muhammad Rizwan | An Efficient Deep Learning Framework for Distracted Driver Detection | Identification of EfficientDet-D3 as the best model for detecting distracted driver. | Efficient Det model detects the objects and the region of interest of the body parts from the images. | Mean Average Precision (MAP) of 99.16% | Model can be improved by using dynamic data. |
| 4. | 2022 | [Muhammad Saiful Haqem Saiful Bahari](https://ieeexplore.ieee.org/author/37089409015), [Lucyantie Mazalan](https://ieeexplore.ieee.org/author/38570550200) | Distracted Driver Detection Using Deep Learning | Images of distracted drivers identified along with a tag to represent a distracted driver in a video. | Pre-processed images by resizing to make them suitable for use with the ResNet-50 model. | 94% accuracy | Need of better image preprocessing technique |
| 5. | 2019 | [Munif Alotaibi](https://link.springer.com/article/10.1007/s11760-019-01589-z#auth-Munif-Alotaibi-Aff1) , Bandar Alotaibi | Distracted driver classification using deep learning | This research investigates distracted driver posture recognition as a part of the human action recognition framework | Proposed a method that combines 3 models, residual network, Inception module and the HNN to improve the performance. | 96.23% accuracy on State Farm dataset and 92.36% accuracy on AUC dataset. | Dataset can be extended to include more classes of distraction. |
| 6. | 2018 | Yehya Abouelnaga, Hesham M. Eraqi, Mohamed N. Moustafa | Real-time Distracted Driver Posture Classification | Presented a new dataset for “distracted driver” posture estimation. | A weighted ensemble of classifiers using a genetic algorithm was created and classification by means of face and hand localizations was done. | 94.29%  classification accuracy which can operate in real-time environment. | Need of performance improvement due to temporal features. |
| 7. | 2020 | Nikka Mofid, Jasmine Bayrooti, Shreya Ravi | Keep Your AI-es on the Road: Tackling Distracted Driver Detection with Convolutional Neural Networks and Targeted Data Augmentation | Built a robust multi-class classifier to detect and identify different forms of driver inattention using the State Farm Distracted Driving Dataset. | Their model combined several augmentation techniques, including skin segmentation, facial blurring, and classical augmentation techniques for the detection of distracted driver. | F1 score of 0.662. | ResNet-50 and ensemble methods performed better than this model. |
| 8. | 2016 | Yuan Liao, Shengbo Eben Li, Wenjun Wang, Ying Wang, Guofa Li, Bo Cheng | Detection of Driver Cognitive Distraction: A Comparison Study of Stop-Controlled Intersection and Speed-Limited Highway | This research focused on “Cognitive driver distraction” and used SVM to extract patterns from noisy data. | Employed SVM-RFE for dimensionality reduction and SVM classifier model was designed for cognitive distraction. | Average accuracy of 81.1% | Improving model accuracy. |
| 9. | 2023 | Abeer. A. Aljohani | Real-time driver distraction recognition: A hybrid genetic deep network-based approach | Proposed framework can be used as a real time driver’s distraction detection to decrease car [traffic accidents](https://www.sciencedirect.com/topics/engineering/highway-accidents) and alleviate corresponding damages to the drivers. | Built a model with combination of [CNN](https://www.sciencedirect.com/topics/engineering/convolutional-neural-network) and  [GA](https://www.sciencedirect.com/topics/engineering/genetic-algorithm) algorithm | 99.80% accuracy | Need of improvement with more data. |
| 10. | 2022 | Tahir Abbas,Syed Farooq Ali,Mazin Abed Mohammed,Aadil Zia Khan,Mazhar Javed Awan,Arnab Majumdar,Orawit Thinnukool | Deep Learning Approach Based on Residual Neural Network and SVM Classifier for Driver’s Distraction Detection | This research proposed ReSVM on 4 datasets namely,  State Farm Distracted Driver Detection, Boston University, DrivFace, and FT-UMT | Introduced ReSVM, an approach combining deep features of ResNet-50 with the SVM classifier, for distraction detection of a driver. | 95.5%  accuracy | A need of alerting system in cars. |

**Chapter 3**

**Software Design**

This chapter details the design for the proposed Distracted Driver Detection with Prioritized Alerting system. The design elements are structured to ensure robust and efficient implementation. This section includes various UML diagrams to visualize the system's architecture, behavior, and interactions. The UML diagrams included are Class Diagrams, Interaction Diagrams (Sequence and Collaboration diagrams), Object Diagrams, Use Case Diagrams, and Control Flow Diagrams. Each diagram serves a specific purpose and collectively provides a comprehensive view of the system's design.

**3.1 UML Diagrams of Proposed Model**

Unified Modeling Language (UML) diagrams are visual representations used to model the structure and behavior of a software system. They provide a standard way to visualize the design of a system, making it easier to understand, document, and communicate its architecture and interactions. Below are explanations of the different types of UML diagrams included in this chapter:

**3.1.1 Class Diagram**

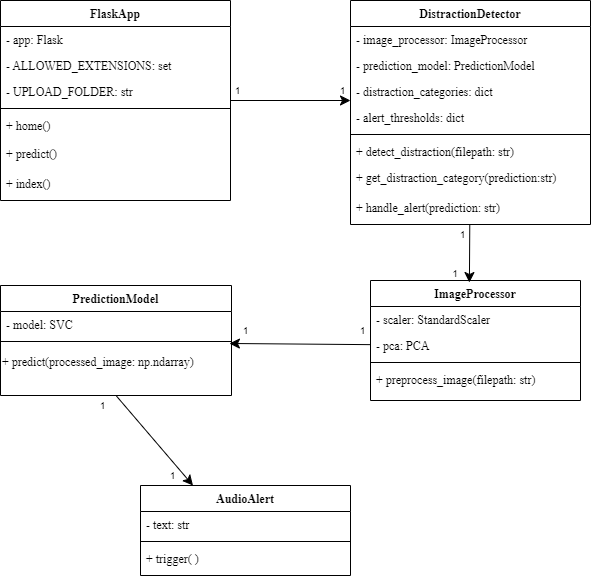
This section outlines the relationship between the classes in the Flask web application as shown in Fig. 3.1.

**FlaskApp and DistractionDetector** classes are linked to each other. The Flask routes, particularly the ‘/predict’ route, interact with the “DistractionDetector” instance to handle the image processing, prediction, and alert generation. The “DistractionDetector” is initialized with the instances of “ImageProcessor” and “PredictionModel”.

**DistractionDetector uses ImageProcessor** classto preprocess the uploaded images. The “preprocess\_image” method in “ImageProcessor” resizes, flattens, scales, and applies PCA transformation to the image to prepare it for prediction.

**DistractionDetector depends on PredictionModel** to make predictions on the pre-processed images. The “predict” method in “PredictionModel” uses the pre trained model to classify images.

**DistractionDetector uses the AudioAlert** class, when a distraction is detected (other than the safe driving), the DistractionDetector class uses AudioAlert to generate and play an audio alert. The handle\_alert method in DistractionDetector creates an instance of AudioAlert and triggers the alert.



**Figure 3.1 Class Diagram of Flask web application**

**3.1.2 Object Diagram**

This section outlines the relationship between the objects in the Flask web application. Fig. 3.2 is a visual representation of the object diagram for the proposed model. The instances involved in the object diagram are detailed as follows:

**app**: An instance of the FlaskApp class, representing the Flask application.  
**distraction\_detector**: An instance of the DistractionDetector class, used by app to manage distraction detection.

**image\_processor**: An instance of the ImageProcessor class, used by distraction\_detector for preprocessing images.

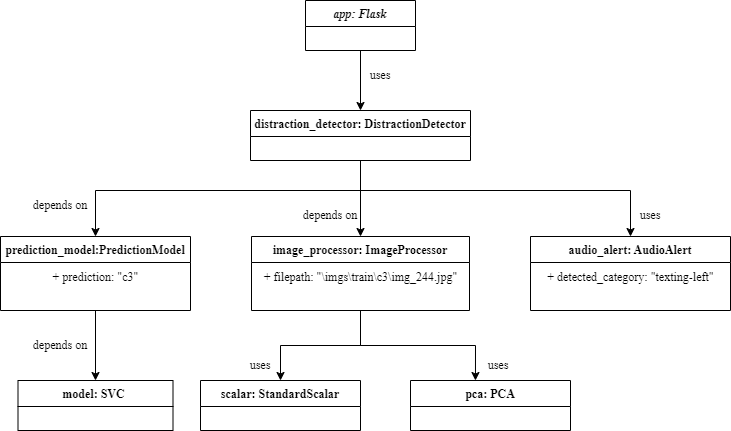
**prediction\_model**: An instance of the PredictionModel class, used by distraction\_detector for making predictions.

**audio\_alert**: An instance of the AudioAlert class, used by distraction\_detector to generate audio alerts when a distraction is detected.

**scaler**: An instance of the StandardScaler class, used by image\_processor for scaling images.

**pca**: An instance of the PCA class, used by image\_processor for performing PCA on images.

**model:** An instance of the Model class, used by prediction\_model to make predictions.



**Fig. 3.2 Object Diagram of Flask web application**

**3.1.3 Use Case Diagram**

In the Use Case Diagram, User is typically a driver or an application user who uploads an image for prediction and System is the backend system which includes the Flask application, the SVM model, and the auditory alerting system. The steps involved in the use case diagram are detailed as follows:

Step 1: User interacts with Upload Image.

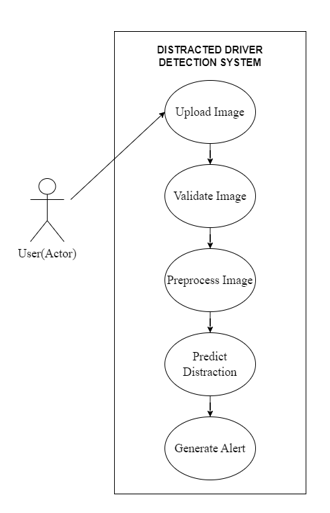
Step 2: Upload Image triggers Validate Image.

Step 3: Upon successful validation, Preprocess Image is executed.

Step 4: Preprocess Image interacts with Predict Distraction.

Step 5: Predict Distraction interacts with Generate Alert if a distraction is detected.

Step 6: Finally, Display Result shows the prediction to the User.

**  
Fig. 3.3 Use Case Diagram of Distracted Driver System**

**3.2 Control Flow Diagram of Proposed Model**

**Fig. 3.4 Control Flow Diagram**



The system workflow for detecting driver distraction using an SVM model with PCA and deploying it with Flask involves several well-defined steps as shown in Fig. 3.4.

Initially, **image data** undergoes preprocessing, including resizing, flattening, and scaling. PCA is then applied to reduce the data's dimensionality, retaining only the most significant features. This step is crucial for minimizing computational load while preserving the data's essential characteristics.

Next, the **SVM model** is configured with specific hyperparameters: a regularization parameter (C=1), a linear kernel, and a one-vs-one decision function shape for multi-class classification. The model is then trained on the pre-processed data, learning to classify images into one of ten distracted driver categories.

After training, the model's performance is **evaluated** using a separate test dataset. Metrics such as accuracy and a confusion matrix are computed to assess how well the model performs.

Once validated, the trained model, along with the scaler and PCA objects, is serialized using pickle and saved as "ddd.pkl" to allow for future reuse without retraining.

In the **deployment** phase, the pretrained model is loaded into a Flask application. This application provides a web interface where users can upload images. The Flask app routes manage various tasks: the main route renders the homepage, while the /predict route manages image uploads. It validates and saves the uploaded files, preprocesses them using the loaded scaler and PCA, and then predicts the distraction category using the SVM model.

To enhance driver safety, an **auditory alerting system** is integrated. When a distraction is detected, an AudioAlert class uses gTTS to convert alert messages to speech and Pygame to play the audio. The severity of the alert is determined by predefined thresholds, ensuring that the warning's intensity matches the detected distraction's risk level.

The entire process, from preprocessing to prediction and alerting, is seamlessly integrated within the Flask application. This ensures that users can easily upload images and receive immediate feedback and alerts, helping to mitigate distracted driving in real-time.

**Chapter 4**

**Requirements and Methodology**

**4.1 Requirements**

**4.1.1 Hardware Requirements**

* RAM - 16 GB
* Processor - Intel Core i7 11th generation (or above)
* Storage - 512GB SSD, 1TB HD
* GPU - Intel Iris(R) Xe GPU

**4.1.2 Software Requirements**

* Operating System - Windows 8/10 (or above)
* Language Used - Python (version- 3.11.4)
* Scikit-learn (version- 1.2.0), Google Text-to-Speech API
* GUI Framework - Flask (version- 2.2.2)
* IDE - Visual Studio Code, Jupyter Notebook

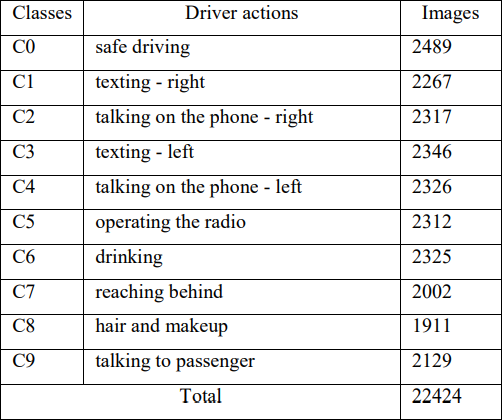
**4.2 Methodology**

**4.2.1 Dataset and proposed classes for dataset extension**

State Farm is an insurance firm in the United States. They surveyed with the help of some subjects, both male and female and created an image dataset which they called the State Farm Distracted Driver Detection (SFDDD) Dataset, depicting different types of distracting activities that drivers tend to indulge in. The dataset consists of images classified into 10 categories, representing nine different types of driver distractions (e.g., texting, talking on the phone, eating) and one category for "safe driving".

The process of getting the system to work begins with gathering data from the dataset. The images are colored and are of size 640×480 pixels. Each class encompasses nearly 2300 images and the distribution per class along with labels is detailed below in Table 4.1. The dataset has been split into train and test subsets. The training subset consists of 10 driver distraction classes (c0- c9) as identified.

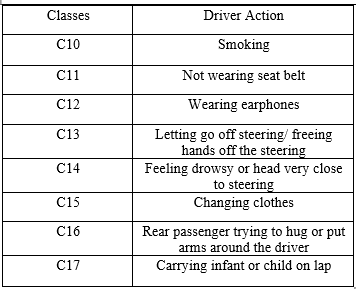
**Table 4.1 Driver Distraction Categories from Dataset**



**Proposed Dataset extension with additional Distraction Classes**

Apart from the classes already available in the State Farm dataset, we have also proposed an extended list of driver distractions that can be incorporated into the database. After a thorough study of traffic-related statistics, we have concluded that there can be many other forms of distraction classes as listed in Table 4.2.

**Table 4.2 Additional Driver Distraction Classes**



Driver distraction is not restricted to the causes enumerated in the preceding pages of this report. There are other factors that need elaborate studies and have been touched in this report as an additional cause effect relationship to the factors that affect the driving and driver safety. Experts strongly recommend wearing seat belts as it not only distracts the drivers but also causes fatalities when the accidents happen. In the 2021 Ministry of Road Transport and Highways Annual report it is mentioned that 16397 people lost lives not wearing seat belts. 7959 out of these were passengers. A total of 39231 were injured for not wearing seat belts. Studies at micro level are yet to be conducted for distraction caused by these factors but the experts in Delhi Traffic Police have shared the expert views that a passenger not wearing the helmet or the seat belt when tossed around in a sudden braking or collision situation dies distract the driver.

Smoking is an activity that distracts a driver significantly. The driver may be searching for the pack, looking for the lighter, lighting while driving, locating the ashtray, or simply put the amber may drop on the person and this may result in sudden loss of control. In a 2007 study - Smoking while driving and its consequences on road safety by L Palumbo and G Mangiaracina it was established that a mobile user while driving is distracted for 10.6 seconds a smoker in comparison loses control for a duration of 12 seconds. While this distraction is only about attention span, the real increased risk is manifold when the increased level of carbon dioxide and Carbon monoxide inside the car leads to impaired control because of the lack of oxygen availability and the brain response in such situation.

Hands free driving is another cause of distraction and accidents occur due to this habit. In the subclass multitasking while driving the National Safety Council Data receives support from the 2010 French study by the Neuroscientists that the brain is not designed to multitask and in fact it constantly switches attention between the tasks. Thus, any activity while driving poses a risk to the road users.

Feeling drowsy while driving is another risk hazard that must be carefully handled and obviated during driving. National High Traffic Safety administration (NHTSA) USA in 2017 estimated that roughly 91000 accidents occurred and 800 people died in road accidents when drivers fell asleep or lost control while driving when they felt drowsy. There are obvious warning signs like Yawning, head falling to the side, missing the signages on the road, feeling nodding off, lane drifting, trouble remembering the last mile marker on the road.

The analysis in the paragraphs indicate that driving skill is closely related to the behaviour on the road. While the factors discussed independently affect driving the composite effect of the activities may be more catastrophic. Multitasking avoidance is a cue from the studies.

**4.2.2 Dataset Visualization and Preprocessing**

In the data visualization and examining phase of our research we first explored the distribution of images in the different classes. There is a total of 10 different driver actions labelled (‘c0’ through ‘c9’). Furthermore, we examined the composition of the dataset, including the division of training images and testing images.

A graph of a number of bars

Description automatically generated

**Figure 4.1 Bar Plot Sampling from Class Distribution**

To visually represent the distribution of classes we utilized data visualization tools. The plot is generated using ‘Seaborn’ and ‘Matplotlib’ libraries. The Matplotlib library is used for 2D plotting of the Class Distribution which includes labelled axes for class names and image counts. Seaborn’s ‘countplot’ function is used to create a bar plot of the class distribution as shown in Fig. 4.1.

Image data is loaded using the ‘skimage’ library, and each image in the dataset is resized to the same dimensions of (150,150,3) using the ‘resize’ function. The resizing of images ensures the uniformity of the input dataset. The pixel values of the resized images were then flattened and stored in an array while corresponding target labels (classes through ‘c0’ to ‘c9’) were stored in another array. Each image is represented by 67,501 features after flattening the pixel values.

For enhanced training, a random sample of the dataset was created which was altered for subsequent model training and evaluation. To improve training efficiency and evaluation accuracy, a random subset of the dataset was generated. This subset comprised 10,000 images, subsequently divided into two sets: a training subset containing 8,000 images and a testing subset containing 2,000 images.

**4.2.3 Dimensionality reduction using Principal Component Analysis (PCA)**

Datasets comprising of images are typically high-dimensional due to the large number of pixels in each image. This high dimensionality not only requires significant storage space but also increases the computational time required for training machine learning models. To mitigate this issue and optimize memory usage PCA was used. Principal Component Analysis is a dimensionality reduction algorithm that identifies principal components, reducing the complexity of dataset while retaining the variance.

In the preprocessing step it was identified that each image represents 67,501 features, these features are further processed based on variance. We utilized ‘sklearn’ library to import PCA from ‘sklearn.decomposition’ module. To perform PCA the standardization of the data is performed using “Standard Scaler”, which involves removing the mean and scaling to unit variance.

The number of principal components selected was 800; we used a plot Fig. 4.2. that shows how much of the variation in the data is explained as we add more components. By looking at where the curve levels off, we could see that 800 components capture approximately 90% of the variance, which encompasses most of the information in the image data. This reduction in dimensionality, from 67,501 to 800, results in more efficient storage and computation. PCA effectively streamlines the dataset to process faster during the model training phase.

A graph with a blue line

Description automatically generated

**Figure 4.2 Number of Principal Components and their cumulative variance**

**4.2.4 Support Vector Machine (SVM) Algorithm**

SVM is a supervised ML algorithm that is used in classification and regression problems. The algorithm uses an input set of data that includes individual weights associated with each element in the dataset. These weights are basically labels.

The output of the algorithm is an optimal hyperplane that helps to separate and classify the dataset points or elements. In a one-dimensional space, the hyperplane is a point, in a 2-D space, the hyperplane is a line, similarly in a 3-D space, the hyperplane is a surface that separates the space into 2 parts where each class of data points lie on either side of the plane.

In the case of 2-D space, we have 2 kinds of hyperplane situations, one is the linearly separable hyperplane, and the other is the non-linearly separable hyperplane, as shown in Fig. 4.3. Space is said to be linearly separated when we can generate a line as a hyperplane. But when the data points are inseparable using a line, it is said to be a non-linearly separated plane.

A diagram of different types of objects

Description automatically generated with medium confidence

**Figure 4.3 Linearly and Non-Linearly Separated Spaces**

There are a lot of possible hyperplanes that can exist in space but what SVM does is, it helps to find the best or the optimal one. The best hyperplane maximizes the margin between the positive and the negative samples. This margin is also called the street around the hyperplane that separates the positive and negative data samples.

The support vectors are the data points that lie closest to the hyperplane and lie on the decision boundaries on either side of the hyperplane. These are the most difficult to find as the position of the optimal hyperplane only depends on the support vectors. Moving the support vectors affects the decision boundaries, but moving other points or vectors does not affect the decision boundaries.

A linear SVM model is selected for its effectiveness in multi-class classification tasks and is trained using the training subset. The instance of the SVM classifier is created with a linear kernel and regularization parameter (‘c’) set to 1.

**4.2.5 Model Training and Approach**

In terms of State Farm dataset, it is evident that the data is divided into multiple classes, indicating an image classification task. SVM can handle the multiclass classification tasks through various strategies such as ‘one vs one’ and ‘one vs rest’. In our proposed framework we have imported the ‘Support Vector Classifier: SVC’ from ‘sklearn’ library, built upon ‘libsvm’. This classifier efficiently addresses multiclass scenarios using the 'one vs one' approach, where binary classifiers are employed pairwise.

In the ‘one vs one’ strategy, a binary classifier is trained on every possible pair of classes. With 'N' classes, the number of binary classifiers trained is given by the formula N(N-1)/2. In our dataset, comprising 10 classes, a total of 45 binary classifiers were employed. We implemented the 'one vs one' strategy in our SVC using the 'decision\_function\_shape' parameter.

The model was trained on a subset of 8,000 using stratified sampling, ensuring that each subgroup's proportion was maintained in both the training and testing sets.

The system diagram of our proposed workflow using SVM classifier and PCA is depicted in Fig. 4.4. The SVM is employed as a classifier to distinguish between various categories of distracted driver actions, specifically identifying whether a driver is distracted or not. Finally, the trained Linear SVM model was evaluated on the test set. Model predictions are compared against the true labels to evaluate the validation accuracy.

A diagram of a computer program

Description automatically generated with medium confidence

**Figure 4.4 System Diagram**

**4.2.6 Implementation of Auditory alerting System**

We have implemented a prioritized alerting system that is based on the PCA supported SVM algorithm. As the PCA supported model generated a better accuracy rate of prediction, we loaded the model onto a pickle file and fed it into our prototype alerting system’s web application.

**Uploading and Predicting the Class**

To create the UI for the alerting system, we have created a basic web application using the flask framework, as shown in Fig. 4.5. The application simply takes any random image as input which belongs to one of the distraction classes from the state farm dataset. It then predicts the class that the image belongs to from using the SVM model. There are 2 options in the application; one is to upload the image of the driver that is to be predicted and the other one is to ‘predict’ the output and then start the warning with the alarm. The system first calls out the warning and then the alarm and then also displays the uploaded image on the screen.

A person driving a car

Description automatically generated

**Figure 4.5 UI of the system**

**Identifying the Risk Levels**

We have also introduced a concept of prioritized alerting onto our model. Prioritized alerting related to the idea of setting the types of distractions onto a hierarchy scale where the auditory alert and warning will be based on the level of risk it poses while driving. The scale is defined using a three-level measure of low, medium, and high risk that a particular distraction poses.

**Table 4.3 Hierarchy scale of fatality levels**

**A table with text on it

Description automatically generated**



In this system, we set different levels of distraction severity thresholds for drivers. We assign numbers (0.1, 0.2, and 0.3) to represent these levels corresponding to the distraction classes as shown in Table 4.3. If a driver's distraction level is below 0.2, it's considered a low risk and triggers a less urgent alert. If the distraction level is between 0.1 and 0.2, it's a bit more serious, so, it triggers a medium-level alert. These alerts help warn the driver about potential dangers on the road. By using these thresholds, the system can better judge how urgent the situation is and give appropriate warnings to keep the driver safe.

**Auditory warnings and Alerting**

To trigger the auditory warning speeches, we have used the ‘gTTS’ or Google Text-to-Speech API to trigger a speech warning and specify the exact risk or type of distraction activity to the driver. The API utilizes multilingual support in the TTS system to provide warnings in different languages, if required. It is followed by an alarm like the one that is triggered when the driver is not wearing the seatbelt.

The difference between a normal alarming system and ours is that our alarming system takes priority or measure of the risk level in consideration and based on that it triggers a mild, a bit strong, and a heavy alarm tone. One issue of using the ‘gTTS’ API is that it requires a stable internet connection to work. This is causing a delay from the time of detecting the distraction to the point of calling out a warning. But the delay is very minimal, which may not affect the aim of the system to alert the driver. The delay ranges from 0.4-0.9 seconds. Also, a conditional check is done to ensure that the prediction is not equal to "c0" since c0 class corresponds to “Safe Driving” for which alert is not required.

**Chapter 5**

**Coding/ Code Templates**

This section consists of essential files and code templates used throughout the project and gives a detailed explanation of folder structure and various classes, their functionalities, and methods with input and output parameters.

**5.1 Directory Structure**

The organization of the project’s folder structure is critical for maintaining clarity and ease of navigation. Fig. 5.1 gives a comprehensive overview of the directory layout.

A screenshot of a computer program

Description automatically generated

**Fig. 5.1 Directory Structure**

**5.2 Directory Descriptions**

**5.2.1 `/App` directory**

The App folder serves as the root directory for the project. It is divided into various files and subdirectories, each serving a specific purpose. The structure of the `App` folder is explained in Fig. 5.1. The files located in this folder are:

**DDD\_Training.py -** This script is designed to train a machine learning model for classifying distracted driver behaviours based on images. It begins by defining the categories of distracted driving and specifying the directory containing the training images.

The script preprocesses the image data by resizing and flattening the images, storing them along with their corresponding labels. The dataset is then split into training and testing sets, and the input data is standardized using StandardScaler and transformed using PCA for dimensionality reduction as shown in Fig. 5.2. A linear Support Vector Classifier (SVC) model is trained on the transformed training data, and its accuracy is evaluated on the test set.

A white background with black text

Description automatically generated  
**Fig. 5.2 Code Snippet of instances ‘scalar’ and ‘pca’**

Finally, the trained model, along with the scaler and PCA components, is saved to a pickle file **model\_data.pkl** for future use in a deployed application as shown in Fig. 5.3.

A screenshot of a computer program

Description automatically generated  
**Fig. 5.3 Code Snippet of trained model**

**app.py** - This file is a Flask web application designed to predict driver distraction categories using a pre-trained machine learning model. It consists of four main classes: ImageProcessor, PredictionModel, AudioAlert, and DistractionDetector. Each class is responsible for a specific part of the process, from preprocessing the image to triggering an auditory alert.

**ImageProcessor** class preprocesses the input image for prediction. The \_\_init\_\_ method takes two input parameters named “scalar” and “pca”, which represents an instance of StandardScalar and PCA class respectively. These instances are loaded from the DDD\_Training script to ensure that the preprocessing steps performed during training are replicated for new input data in the Flask application, maintaining consistency and accuracy in predictions.

The preprocess\_image(filepath) function reads the image from the given file path, resizes it to 150x150 pixels, flattens it, scales it using the StandardScaler, and applies PCA transformation.

A screenshot of a computer program

Description automatically generated  
**Fig. 5.4 Code Snippet of ImageProcessor class**

**PredictionModel** class predicts the distraction category of the pre-processed image. The \_\_init\_\_ method takes “model” as the input parameter, which represents the instance of pre trained model loaded from DDD\_Training script. The predict () function takes the pre-processed image as the input, specified as a NumPy array, and returns the predicted distraction category.

A computer code with black text

Description automatically generated  
**Fig. 5.5 Code Snippet of PredictionModel class**

**AudioAlert** class converts a given alert message generated using the predicted distracted category into speech and plays it as an alert. The trigger() method uses the gTTS library to convert text to speech, saves it as an mp3 file, and uses pygame to play the audio.

A computer screen shot of text

Description automatically generated  
**Fig. 5.6 Code Snippet of AudioAlert class**

**DistractionDetector** class integrates the functionalities of “ImageProcessor” and “PredictionModel” class to detect distraction and handle alerts. The \_\_init\_\_ method initializes the DistractionDetector object with three parameters: "model," "scaler," and "pca." These parameters represent the pre-trained machine learning model, StandardScaler instance, and PCA instance, respectively, loaded from the DDD\_Training.py script. It creates an “ImageProcessor” object named "image\_processor" using the provided scaler and pca instances and a “PredictionModel” object named "prediction\_model" using the provided model instance. It also initializes dictionaries "distraction\_categories" and "alert\_thresholds" containing distraction categories and corresponding alert thresholds, respectively as shown in Fig. 5.7.

A screenshot of a computer program

Description automatically generated  
**Fig. 5.7 Code Snippet of DistractionDetector class**

The detect\_distraction(filepath) function processes the image and predicts the distraction category.

A computer code with black text

Description automatically generated  
**Fig. 5.8 Code Snippet of detect\_distraction( ) function**

The get\_distraction\_category(prediction) function returns the name of the distraction category which is further used in index.html to display the predicted category class.

A black and orange text

Description automatically generated  
**Fig. 5.9 Code Snippet of get\_distraction\_category( ) function**

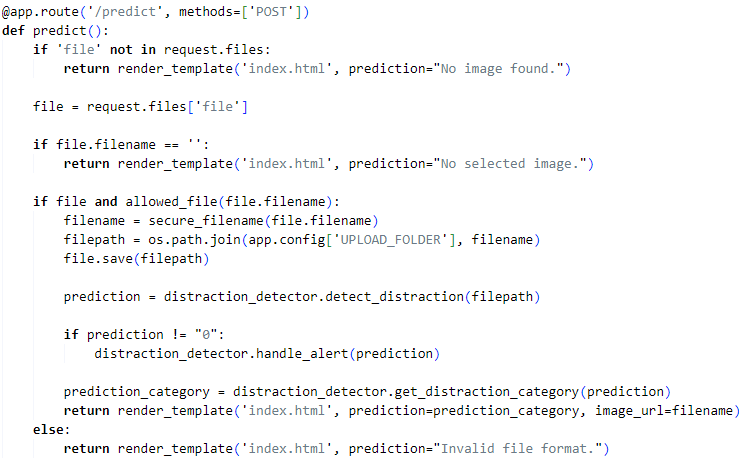
The handle\_alert(prediction) triggers an audio alert if the predicted category indicates a distraction. Manages alert priority based on predefined thresholds.

**A computer code with white text

Description automatically generated  
Fig. 5.10 Code Snippet of handle\_alert( ) function**

Once the distraction category is determined, an alert message is generated to notify the driver about the detected distraction. This message includes information about the specific distraction category detected by the model. After generating the alert message, an audio alert is triggered using the ‘pygame’ library. This library allows for the playback of audio files in Python applications. The selection of audio files for the alert is based on the severity of the distraction category detected. Different audio files are used for low, medium, and high priority distractions, providing varying levels of urgency in the alert. The selected audio files are loaded and played using pygame.mixer.music.load() and pygame.mixer.music.play() functions respectively. These functions initiate the playback of the audio files in the background. The alert continues to play until the audio file finishes or is manually stopped.

The Flask Application “app.py” defines several routes for handling user requests and responses related to distraction detection. These routes are used to define the behavior of the web application when accessed through different URLs. Home Page Route (/) renders the home page “home.html”, Upload Page Route (/home) renders the image upload page “index.html”, Prediction Route (/predict) handles image uploads, predicts distraction categories, and triggers alerts if necessary.

  
**Fig. 5.11 Code Snippet of Prediction Route (/predict)**

**5.2.2 `/imgs` directory**

The “imgs” folder consists of the dataset taken from Kaggle “State Farm Distracted Driver Detection Dataset”. There are two subdirectories named "train" and "test" each containing images for training and testing purposes, respectively. Within the "train" directory, there are ten subdirectories, each representing a different class or category. These subdirectories serve to organize the images based on their corresponding classes. Each class directory contains multiple images associated with that category.

**5.2.3 `/static` directory**

It is a directory to store static files, including uploaded images. The static folder is a standard convention in Flask applications, and it helps in organizing the project's assets.

**5.2.4 `/templates` directory**

It is a directory containing HTML templates for rendering web pages. The “index.html” file located in the “templates” directory serves as the user interface for the Flask web application.

A screenshot of a computer program

Description automatically generated  
**Fig. 5.12 Code Snippet of form in index.html**

The html page provides a form for uploading an image as shown in Fig. 5.12. When submitted, the form sends a POST request to the "/predict" route of the Flask application. **Input Element** allows users to select an image file from their device, **Submit Button** initiates the upload and prediction process when clicked. If a prediction is available (i.e., after the image has been uploaded and processed), it is displayed as a heading below the form. Additionally, the uploaded image is displayed below the prediction.

**5.2.5 `/TrainingPhase` directory**

In the training phase, Jupyter notebooks were instrumental in evaluating the machine learning model's performance and determining the training latency. Two notebooks, namely "DDD\_withPCA" and "DDD\_withoutPCA" were employed for comparative analysis. These notebooks facilitated the assessment of model performance with and without Principal Component Analysis (PCA), allowing for a comparison of classification accuracy and training time. Several evaluation metrics were employed, including the confusion matrix, which provides insight into the model's classification performance.

**Chapter 6**

**Testing**

Testing of a machine learning model without scrutinizing the code, technology, and internal architecture is called **black box testing**. It focuses on the system’s external behavior and the input-output relationships.

In the Distracted Driver Detection system, input is an image and expected output is a distraction category along with prioritized alerting. Prioritized alerting is based on the level of fatality defined by high risk, medium risk, and low risk. If the provided image is found to be in the category of high risk, an appropriate alert tone and message must be conveyed to the driver.

We have implemented testing on the Distracted Driver System in two ways as described below:

**6.1 Model evaluation library: scikit-learn**

The scikit-learn library in python is used on testing subset to evaluate the model performance by comparing the predicted and expected output.

The Distracted Driver Detecting system correctly classifies 99% of the images which are obtained from the comparison of true labels and predicted labels, the count of correct predictions determined the accuracy of the system. Fig. 6.1 shows the predicted and original label for a testing subset of 2000 images.

A screenshot of a computer program

Description automatically generated  
**Fig 6.1 Comparison of original and predicted labels**

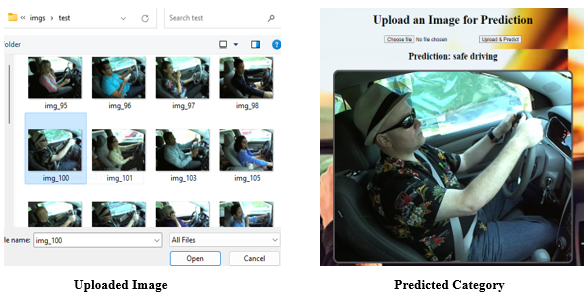
**6.2 Manual Testing**

The implementation of manual testing interface (black-box testing) ensures that the distracted driver system not only categorizes distraction accurately but also issues the correct prioritized alerts based on fatality level. The use of manual testing in the system helps in identifying how the system responds to expected and unexpected user actions, its response time and usability issues. The test data consists of normal and edge test cases.

**6.2.1 Normal Test Cases**

**Test Case 1**: Safe Driving (No risk)

In this test case an image categorized under "safe driving" (class "0") is uploaded using the provided web interface as shown in Fig. 6.2. The system accurately classifies this image as "safe driving" and did not generate any alerts.



**Fig 6.2 Normal Test Case 1 (Safe Driving)**

**Test Case 2**: Reaching Behind (high risk)

In this test case an image categorized under "reaching behind" (class "7") is uploaded using the provided web interface as shown in Fig. 6.4. The system accurately classifies this image as "reaching behind" and identifies the level of risk, producing alerts with minimal latency.

A close up of a message

Description automatically generated  
**Fig 6.3 Fatality Level Identified and Alerting Latency**

**A screenshot of a computer

Description automatically generated  
Fig 6.4 Normal Test Case 2 (Reaching Behind)**

**Test Case 3**: Talking on the phone - left (medium risk)

In this test case an image categorized under "Talking on the phone - left" (class "4") is uploaded using the provided web interface as shown in Fig. 6.5.

A screenshot of a screenshot of a person in a car

Description automatically generated  
**Fig 6.5 Normal Test Case 3 (Talking on the phone – left)**

The system accurately classifies this image as "Talking on the phone -left" and identifies the level of risk, producing alerts with minimal latency as shown in Fig. 6.6.

A screenshot of a phone

Description automatically generated  
**Fig 6.6 Fatality Level Identified and Alerting Latency**

**Test Case 4**: Operating the radio (low risk)

In this test case an image categorized under "Operating the radio" (class "5") is uploaded using the provided web interface as shown in Fig. 6.7.

**A screenshot of a computer

Description automatically generated  
Fig 6.7 Normal Test Case 4 (Operating the radio)**

The system accurately classifies this image as "Operating the radio" and identifies the level of risk, producing alerts with minimal latency as shown in Fig. 6.8.

A screenshot of a computer

Description automatically generated  
**Fig 6.8 Fatality Level Identified and Alerting Latency**

**6.2.2 Edge Test Cases**

**Test Case 1:** Invalid Format of file

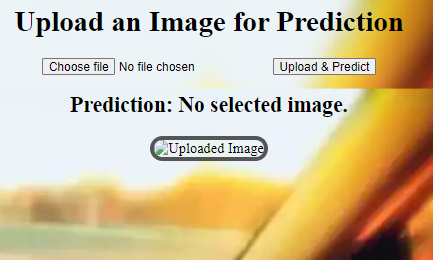
**A screenshot of a computer screen

Description automatically generated  
Fig 6.9 Edge Case 1 (Invalid file)**

In this test case a pdf file is uploaded using the provided web interface. The system provides an error message “Invalid file format” successfully as shown in Fig. 6.9.

**Test Case 2:** Nothing Uploaded

In this test case the “Upload & Predict” button was clicked by the user without uploading any image which resulted in an error message “No selected image” as shown in Fig. 6.10.

  
**Fig. 6.10. Edge Case 2 (Nothing Uploaded)**

**Chapter 7**

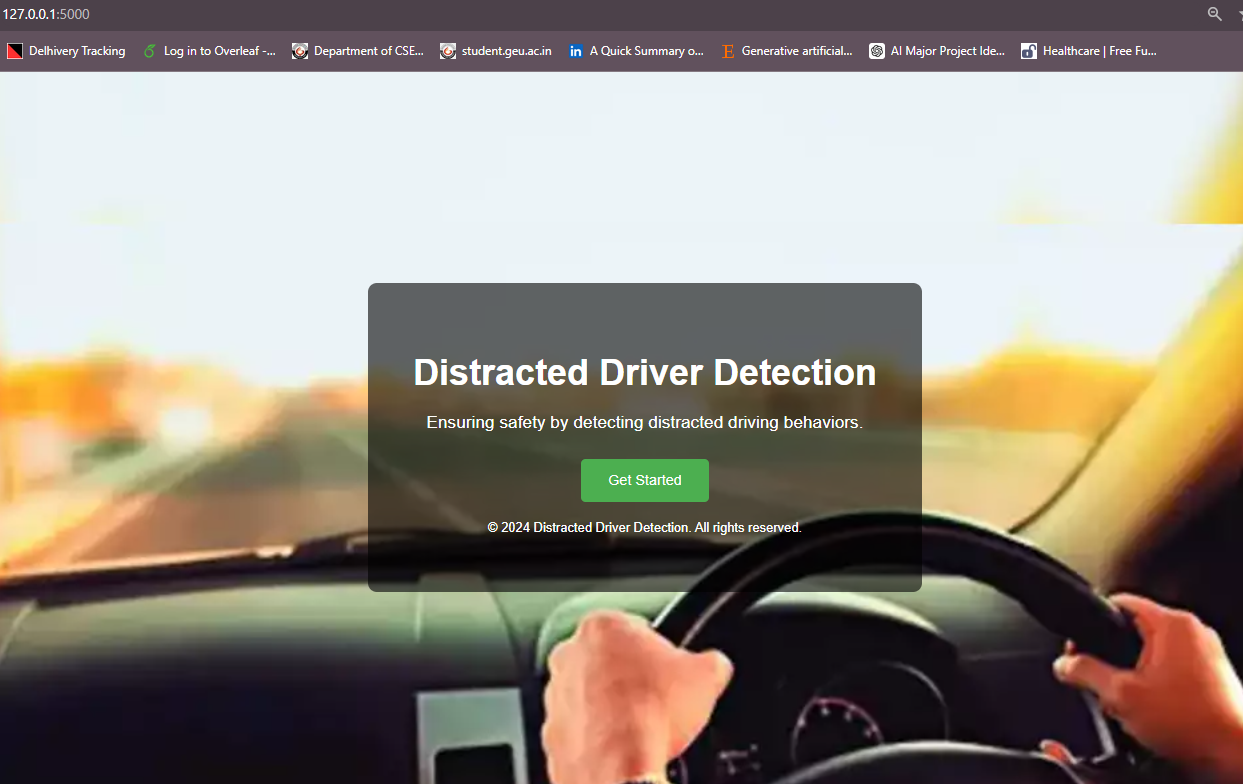
**Results and Discussion**

After the implementation of distracted driver detection model, a flask web application was developed. This model, saved in a pickle file, predicts the category of an input image, and provides prioritized alerting based on the risk level associated with the detected distraction. The application features various user interfaces and output screens to facilitate image upload, prediction display, and alerting mechanisms. This section will discuss the key components and results of the application.

**7.1 User Interfaces and Output Screens**

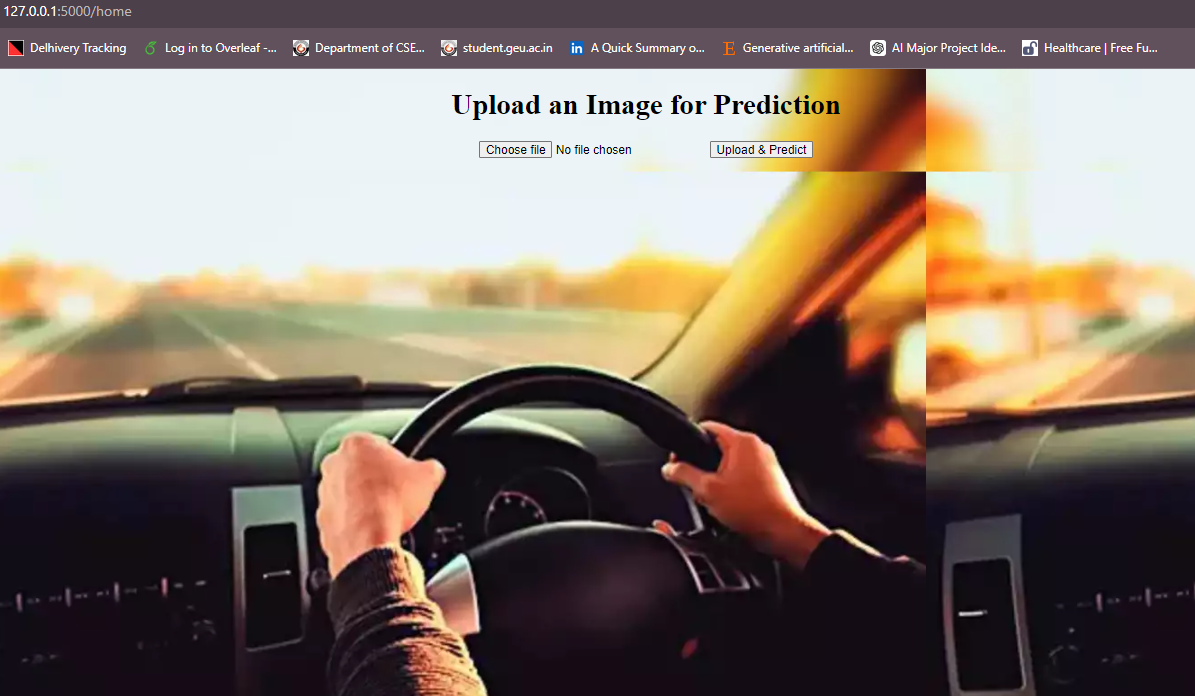
**7.1.1 Home Page**

The home page of the application welcomes the user and provides a way to navigate to the main functionality of the application. The output screen “home.html” can be seen in Fig. 7.1.

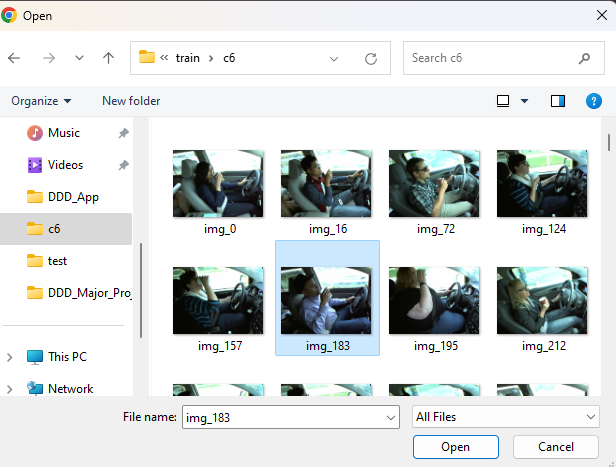
  
**Fig. 7.1 Home page Screenshot**

**7.1.2 Main Page**

The main page prompts the user to upload the image for prediction as shown in Fig. 7.2. The interface includes an **input field** to upload an image file, a **button** to submit the uploaded image for prediction, a **section** to display the prediction results and the uploaded image.

  
**Fig. 7.2 Main page Screenshot**

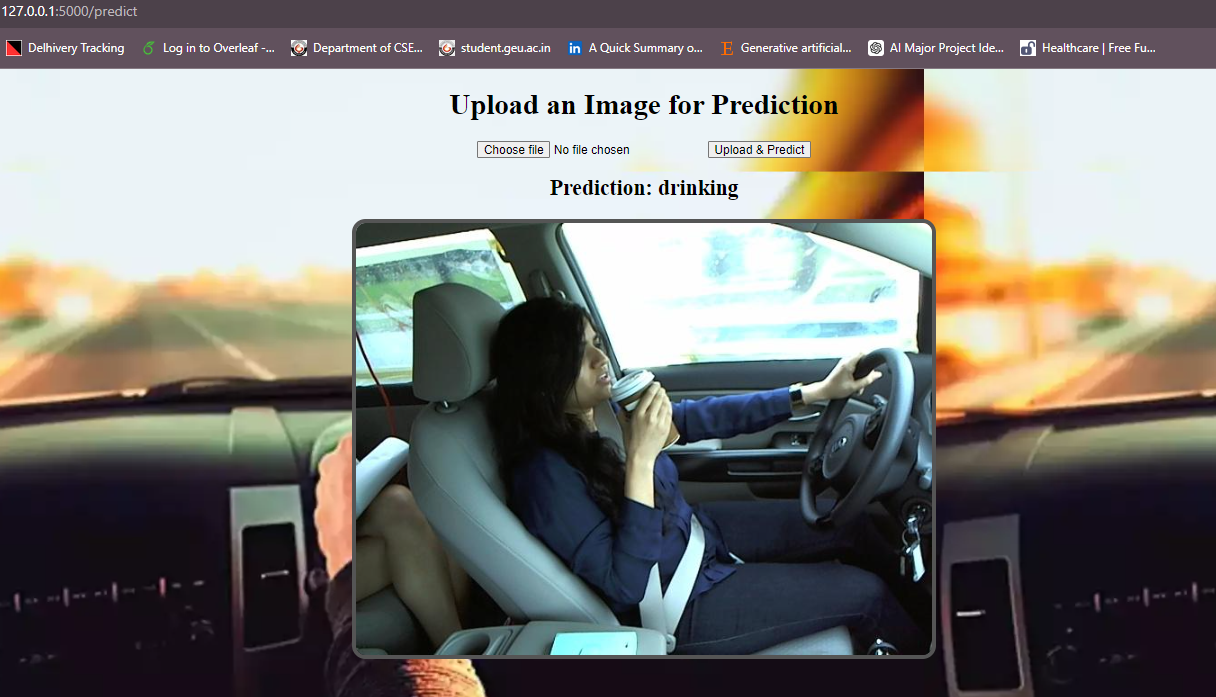
The main window displayed a "Select Image" button and a result label. Clicking the "Select Image" button opened a file dialog that allowed us to choose an image file from any directory as shown in Fig. 7.3.

**  
Fig. 7.3 File Dialog**

**7.1.3 Prediction Output**

After uploading an image, the application processes the image and displays the predicted distraction category along with the uploaded image. If the model detects a distraction, it triggers an alert based on the risk level associated with the detected category.

There are 2 options in the application; one is to upload the image of the driver that is to be predicted and the other one is to ‘predict’ the output and then start the warning with the alarm. The system first calls out the warning and then the alarm and then displays the uploaded image on the screen as shown in Fig.7.4.

  
 **Fig. 7.4 Prediction Output**

When the system automatically identifies the fatality level and plays the warning with the alarm, the “app.py” logs the identified risk and alerting latency using the print statements. Fig. 7.5 shows the output of the print statements.

A close-up of a computer screen

Description automatically generated  
**Fig. 7.5 Fatality Level Identified and Alerting Latency**

**7.2 Model Evaluation and Discussion**

**7.2.1 Confusion Matrix**

The confusion matrix provides insight into the performance of the multiclass classification model on the testing subset of 2000 images as shown in Fig. 7.6. Each row represents the actual class, while each column represents the predicted class. The diagonal elements indicate the number of correctly classified instances for each class, while off-diagonal elements represent misclassifications.

A chart with numbers and a line

Description automatically generated with medium confidence  
**Fig. 7.6 Confusion Matrix for SVM with PCA**

However, our model exhibited confusion primarily between the instances from the Hair and Makeup class (c8), and instances from 5 other classes: Talking to Passenger (c9), Talking on the Phone – Left (c4), drinking (c6), operating the radio (c5), and Safe driving (c0). This confusion may stem from similar visual features shared among these classes, such as hand gestures or head movements, which could be present during both safe driving and distracted behaviors like interacting with passengers or using a phone.

**7.2.2 Comparative Analysis of Performance: PCA vs. Non-PCA-based Approaches**

We conducted our model training and testing with different sets of features. In the initial phase, we implemented an SVM-based framework with 67,501 features, which involved resizing the images to impose uniformity and flattening the pixels into a 1-dimensional array with corresponding labels. Finally, following image preprocessing, we executed a feature reduction step using Principal Component Analysis (PCA), effectively reducing the dimensionality of the image features from 67,501 to 800. Remarkably, the SVM model, operating without dimensionality reduction, achieved a commendable accuracy of 98.95%. However, this came at the expense of a considerable training time of approximately 30 minutes 27 seconds. Upon integrating PCA into our framework, we observed not only an improved model accuracy but also a substantial reduction in training time, now merely 11.2685 seconds. This optimization transformed our framework to be more efficient, exhibiting improved memory utilization and processing speed. A comprehensive comparison of results is shown in Table 7.1.

**Table 7.1 Comparative Analysis of SVM performance**

|  |  |  |
| --- | --- | --- |
| **Metrics** | **Without PCA** | **With PCA** |
| **Model Accuracy** | 98.95% | 99% |
| **Training Time** | 30 min and 27.585141 sec | 11. 268573 sec |
| **No. of Features** | 67501 | 500 |

**Chapter 8**

**Conclusion and Future Work**

**8.1 Conclusion**

This research work has presented a comprehensive study on Driver distraction detection systems using a machine learning based approach. The proposed methodology has demonstrated promising insights and results accurately.

In this project, we addressed the critical issue of detecting distracted driving behaviors using the State Farm Driver Distraction Image Dataset from Kaggle. The dataset consists of images classified into 10 categories, representing nine different types of driver distractions (e.g., texting, talking on the phone, eating) and one category for "safe driving". We adopted a machine learning approach to solve the problem. On studying some papers in the same related field, we found out that two of the best performing algorithms in machine learning were Support Vector Machine (SVM) and Decision Trees. But even then, SVM outperformed the Decision Tree algorithm. So, we worked and implemented Linear SVM algorithm on our dataset achieving an outstanding accuracy of 98.94% from the model. The model training time was roughly 30 minutes 27 seconds. Despite such great accuracy, the tradeoff with time and space was too much. So, to optimize the result to reduce training time and an attempt to enhance accuracy at the same time, we used a dimensionality reduction technique called Principal Component Analysis (PCA). Combining PCA with SVM we reduced the processing components of the dataset from 67501 to 800 and reduced the training time to ~11.2 seconds. More importantly, this increased the accuracy of the model to 99%. The result is remarkable as it is also better than some of the existing research works that use the SVM algorithm.

We also implemented a prototype auditory alerting system which detects the driver’s distraction activity and gives different auditory warnings based on how fatal the distraction is. For example, texting while driving may be classified as high-priority, while talking on the phone might be considered lower-priority. Such hierarchy helped the creation of an auditory alert system. We created a simple application using Flask framework for demonstration purposes.

Furthermore, the project serves as a practical example of applying computer vision and machine learning techniques in the context of real time traffic management systems, showcasing its potential to elevate the level of road safety.

**8.2 Future Work**

As with any research endeavor, there are limitations to consider. The future scope of this research paper offers some avenues for further exploration and improvements. State Farm is a very large dataset and processing of the images requires a lot of time and space, therefore there’s a need for potential optimizations and changes. The categories identified by State Farm Dataset are limited and would fail to identify potential categories that can cause a driver to be distracted from the road, therefore our study proposes 8 new categories that can enhance the classification. Taking multiple subjects to expand the dataset into more diversified classes of distractions can lead to a better model with more accurate predictions.

It is possible to replace the gTTS (Google Text-to-speech) module which we have used to trigger auditory alerts, by a module which can operate offline. This would reduce the delay between detection and alerting as it would not depend upon the internet speed.

One of the most intriguing areas for future application will be focused on improving the computational performance of the model by integrating our existing PCA based SVM model with deep learning or CNN approach. Hybrid approaches used in the model training tend to give better results and implementation.

In conclusion, the future scope areas outlined in this paper offer exciting opportunities for continued research and development, paving the way for advancements in the field of traffic modernization and management and promoting road safety.

**Details of Research Publication**

The status of our research publication are as follows:

R. Amola, S. Bahuguna, R. Kumar, Guru Prasad M.S, H.P. Warkari, "Driver distraction detection system based on SVM and PCA with categorized alerting" 2024 8th International Conference on Computing, Communication, Control and Automation (ICCUBEA-2024) of the IEEE Pune section, Pune, India, 2024 **(Communicated)**

A screenshot of a computer

Description automatically generated

**Figure A. Proof of Communication ICCUBEA**

**References**

1. Furkan Omerustaoglu, C. Okan Sakar, Gorkem Kar, "Distracted driver detection by combining in-vehicle and image data using deep learning," Sensors, vol. 96, p. 106657, November 2020.
2. Monagi H. Alkinani, Wazir Zada Khan, Quratulain Arshad, Mudassar Raza, "HSDDD: A Hybrid Scheme for the Detection of Distracted Driving through Fusion of Deep Learning and Handcrafted Features," Sensors, vol. 22, Issue:5, p. 1864, February 2022.
3. Deepthi M. Pisharody, Binu P. Chacko, K.P. Mohamed Basheer, "Driver distraction detection using machine learning techniques," Materials today, Proceedings, vol. 58, part 1, pp. 251-255, February 2022.
4. Y. Yao, X. Zhao, H. Du, Y. Zhang and J. Rong, "Classification of distracted driving based on visual features and behavior data using a random forest method" in Transportation Research Record: J. Trans. Res. Board, Sep. 2018.
5. K. Sengupta, A. Srivastava, S. Shreyansh, S. Aggarwal, and V. N. Kumar, 'Driver sleep detection: A new and accurate approach,' in Proc. Innov. Power Adv. Comput. Technol. (i- PACT), pp. 1–7, 2021.
6. K Sudheevar Reddy, Kiran N Deshmukh, Lokesh M, Prathik Raj RC, Dr. Kavitha KS, "Deep Learning Approaches to Detect Distracted Driver Activities: A Survey", International Journal for Research in Applied Science & Engineering Technology, vol. 11, Issue:2, pp. 1074-1082, February 2023.
7. T. Liu, Y. Yang, G. -B. Huang, Y. K. Yeo and Z. Lin, "Driver Distraction Detection Using Semi-Supervised Machine Learning," in IEEE Transactions on Intelligent Transportation Systems, April 2016 .
8. K. Seshadri, F. Juefei-Xu, D. K. Pal, M. Savvides, and C. P. Thor, "Driver cell phone usage detection on strategic highway research program (SHRP2) face view videos," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW), pp. 35–43, Jun. 2015.
9. K. Roy, "Unsupervised sparse, nonnegative, low rank dictionary learning for detection of driver cell phone usage," IEEE Trans. Intell. Transp. Syst., vol. 23, no. 10, pp. 18200– 18209, Oct. 2022.
10. Yaman Albadwai, Maen Takruri and Mohammed Awad, "A Review of Recent Developments in Driver Drowsiness Detection Systems", 2022.
11. Yulan Liang, Michelle L. Reyes and John D Lee, "Real-Time Detection of Driver Cognitive Distraction Using Support Vector Machines," IEEE Transactions on Intelligent Transportation Systems, vol. 8, Issue:2, pp. 340-350, July 2007.
12. Y. Xing, C. Lv, H. Wang, D. Cao, E. Velenis, and F. Y. Wang, "Driver activity recognition for intelligent vehicles: A deep learning approach," IEEE Trans. Veh. Technol., vol. 68, no. 6, pp. 5379–5390, June 2019.
13. Muhammad Saiful Haqem Saiful Bahari, Lucyantie Mazalan, "Distracted Driver Detection Using Deep Learning", Proc. IEEE 18th International Colloquium on Signal Processing & Applications (CSPA), May 12-12, 2022.
14. Abeer. A. Aljohani, "Real-time driver distraction recognition: A hybrid genetic deep network-based approach," Alexandria Engineering Journal, vol. 66, pp. 377-389, March 2023.
15. J. Hu, "Automated detection of driver fatigue based on AdaBoost classifier with EEG signals,' Frontiers Comput. Neurosci., vol. 11, p. 72, Aug. 2017.
16. F. Tango and M. Botta, "Real-time detection system of driver distraction using machine learning," IEEE Trans. Intell. Transp. Syst., vol. 14, no. 2, pp. 894–905, June 2013.
17. H. M. Eraqi, Y. Abouelnaga, M. H. Saad, and M. N. Moustafa, "Driver distraction identification with an ensemble of convolutional neural networks,' J. Adv. Transp., vol. 2019, pp. 1–12, Feb. 2019.
18. M. D. Hssayeni, S. Saxena, R. Ptucha, and A. Savakis, 'Distracted driver detection: Deep learning vs handcrafted 2017.
19. D. Tran, H. Manh Do, W. Sheng, H. Bai, and G. Chowdhary, "Real-time detection of distracted driving based on deep learning," IET Intell. Transp. Syst., vol. 12, no. 10, pp. 1210–1219, Dec. 2018.
20. H. V. Koay, J. H. Chuah, and C.-O. Chow, 'Convolutional neural network or vision transformer? Benchmarking various machine learning models for distracted driver detection,' in Proc. IEEE Region 10 Conf. (TENCON), pp. 417–422, Dec. 2021.
21. Prof. Pramila M. Chawan, Shreyas Satardekar, Dharmin Shah, Rohit Badugu, Abhishek Pawar, "Distracted Driver Detection and Classification," Int. Journal of Engineering Research and Application, vol. 8, Issue:4 (Part -III), pp. 51- 55, April 2018.
22. Alexey Kashevnik, Roman Shchedrin, Christian Kaiser, Alexander Stocker, "Driver Distraction Detection Methods: A Literature Review and Framework," IEEE Access, vol. 9, pp. 60063 – 60076, April 2021.
23. Faiqa Sajid, [Abdul Rehman Javed,](https://ieeexplore.ieee.org/author/37088389492) Asma Basharat, Natalia Kryvinska, Adil Afzal, Muhammad Rizwan, "An Efficient Deep Learning Framework for Distracted Driver Detection," IEEE Access, vol. 9, pp. 169270-169280, December 2021.
24. Munif Alotaibi, Bandar Alotaibi, "Distracted Driver classification using deep learning," Signal, Image and Video Processing, vol. 14, pp. 617-624, November 2019.
25. Yehya Abouelnaga, Hesham M. Eraqi, Mohamed N. Moustafa, "Real-time Distracted Driver Posture Classification," NIPS Workshop on Machine Learning for Intelligent Transportation Systems, December 2018.
26. Nikka Mofid, Jasmine Bayrooti, Shreya Ravi, "Keep Your AI-es on the Road: Tackling Distracted Driver Detection with Convolutional Neural Networks and Targetted Data Augmentation," June 2020, arXiv preprint arXiv:2006.10955.
27. Achmad Yuneda Alfajr1, Kartini, Anggraini Puspita Sari, "Classification of distracted driver using support vector machine based on principal component analysis feature reduction and convolutional neural network," Jurnal Informatika dan Komputer, vol. 11, No. 2, pp. 237-245, October 2023.