MANTHRA X : PIONEERING PRECISION, THE FUTURE OF AUTONOMOUS MOBILITY

Akalanka P.A.A - IT21160448
Ganepola G.A.N.B. - IT21155048
Athukorala W.A.A.D.D.T - IT21162978
Dissanayaka D.M.M.I.T - IT21174780

BSc (Hons) degree in Information Technology Specializing in Data Science

Department of Information Technology

Sri Lanka Institute of Information Technology
Sri Lanka

April 2025

MANTHRA X : PIONEERING PRECISION, THE FUTURE OF AUTONOMOUS MOBILITY

Akalanka P.A.A - IT21160448
Ganepola G.A.N.B. - IT21155048
Athukorala W.A.A.D.D.T - IT21162978
Dissanayaka D.M.M.I.T - IT21174780

Dissertation submitted in partial fulfillment of the requirements for the Bachelor of Science (Hons) in Information Technology Specializing in Data Science

Department of Information Technology

Sri Lanka Institute of Information Technology
Sri Lanka

April 2025

DECLARATION

We declare that this is our own work and this dissertation1 does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any other university or institute of higher learning and to the best of our knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgment is made in the text.

Also, we hereby grant to Sri Lanka Institute of Information Technology, the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. We retain the right to use this content in whole or part in future works (such as articles or books).

Name	Student ID	Signature
Akalanka P.A.A	IT21160448	[illegible]
Ganepola G.A.N.B.	IT21155048	[illegible]
Athukorala W.A.A.D.D. T	IT21162978	[illegible]
Dissanayaka D.M.M.I.T	IT21174780	[illegible]

The above candidate has carried out research for the bachelor's degree. Dissertation under my supervision.

Signature of the supervisor: Date: 2025-04-11

ACKNOWLEDGEMENT

We extend my deepest gratitude to my dissertation supervisor, Mr. Samadhi Rathnayake, for his invaluable guidance, constructive feedback, and constant encouragement throughout this research. His expertise in autonomous systems and patient mentorship were instrumental in shaping the direction and execution of this work. We are equally thankful to my co-supervisor, Dr. Lakmini Abeywardhana, for her rigorous academic insights, technical advice, and support in navigating complex challenges.

We acknowledge the Faculty of Computing at the Sri Lanka Institute of Information Technology (SLIIT). This project would not have been possible without the relentless efforts of our dedicated group members. Our teamwork and shared commitment to excellence turned challenges into opportunities. Finally, We thank our parents, siblings, and friends for their emotional support, patience, and belief in our abilities during the demanding phases of this research. Their encouragement kept us motivated to strive for quality and innovation.

ABSTRACT

Advancements in autonomous vehicle technology have raised new issues about safety, smart, and morally correct decision making in ever-changing real-world driving conditions. The present report proposes MANTHRA-X as an end-to-end solution for enhancing the safety and operating effectiveness of autonomous vehicles. MANTHRA-X combines advanced concepts in deep reinforcement learning (DRL), ethics based decision making, perception, and in cabin security in a unified platform. Core modules include Perception and Scene Understanding for real-time object detection and motion prediction, Decision Making and Collision Avoidance using hybrid control strategies such as Deep Q Network (DQN) and PID controller, and an Ethical Decision-Making module utilizing object prioritization and driver emotions to guide responsible behavior. An alerting system and simulation interface also provide real-time feedback to users. The modularity-oriented architecture and real-time responsiveness are aimed at delivering robust performance in uncertain and hostile driving situations. The MANTHRA-X system has been demonstrated to exhibit positive results in improving situational awareness, reducing collision rates, and overall decision-making capacity in dynamic environments under strict simulations and test schedules. The findings of this project are a beneficial addition to the nascent area of autonomous systems by proving how smart fusion of DRL, ethical choice, and sensor-based perception can lead to more adaptive, safer, and context-aware autonomous vehicles.

Key words: *Autonomous Vehicles, Deep Reinforcement Learning, Hybrid Control Systems*

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

ABBREVIATION

Abbreviation	**Full meaning**
AV	Autonomous Vehicle
CNN	Convolutional Neural Network
DRL	Deep Reinforcement Learning
DQN	Deep Q-Network
GNN	Graph Neural Network
PID	Proportional-Integral-Derivative (Controller)
YOLOv5/v8	You Only Look Once (Object Detection Model)
mAP	Mean Average Precision
RMSE	Root Mean Square Error
FPS	Frames Per Second
IoT	Internet of Things
API	Application Programming Interface
UI	User Interface
UAT	User Acceptance Testing
CARLA	CAR Learning to Act (Simulator)

1 INTRODUCTION

MANTHRA-X: Pioneering Precision, The Future of Autonomous Mobility is a groundbreaking research project aimed at addressing some of the most critical challenges in the development of autonomous vehicles (AVs). While AVs are hailed as a transformative technology, significant hurdles remain in their ability to become fully reliable, safe, and seamlessly integrated into society. These challenges are particularly evident in Level 3 autonomous vehicles, where the vehicle can operate autonomously under certain conditions, but the human driver must remain alert and ready to intervene when required.

The rapid advancements in artificial intelligence (AI), machine learning, sensor technologies, and real-time data processing have catalyzed the development of autonomous vehicles. However, despite these advancements, AVs are still far from perfect. Current systems often fail to integrate essential components into a cohesive support system that ensures the safety, reliability, and ethical decision-making necessary for autonomous driving.

MANTHRA-X is an innovative solution developed to address these shortcomings. It integrates a wide range of technologies and methodologies into a unified, adaptive system that enhances the operational performance of Level 3 autonomous vehicles. MANTHRA-X integrates the following components:

1. Object Detection and Motion Prediction: Using advanced machine learning and sensor fusion techniques, MANTHRA-X enables real-time detection and tracking of surrounding objects, ensuring the vehicle can identify obstacles and predict their motion with high accuracy, even in complex and dynamic driving conditions.

2. Decision Making and Collision Avoidance: MANTHRA-X incorporates a hybrid decision-making framework merging Deep Q-Networks (DQN) with conventional PID control to enhance collision avoidance for autonomous vehicles. The DQN component is responsible for high-level decision-making, learning the best driving policies such as lane changing, braking, or evasive maneuvers from sensor inputs in real-time. Inputs are LiDAR, RGB cameras,

and GPS, fused to provide a global understanding of the environment. Once the decision is reached, smooth and stable action implementation is enabled by the PID controller via regulation of steering, throttle, and brakes. There is a real-time collision avoidance and alerting system integrate with this. When there is a perceived threat, it can override DQN's decision to safeguard. This multi-layer strategy enables the system to respond sensibly and responsively to intricate traffic conditions, greatly reducing collision risks and improving general road safety.

3. Ethical Decision Making and Driver Awareness Monitoring: In situations where clear-cut decisions are not possible, MANTHRA-X incorporates an ethical decision-making framework that allows the vehicle to make morally sound choices in unavoidable accident scenarios. Additionally, MANTHRA-X includes a Driver Awareness Monitoring system, which ensures the driver remains alert and ready to take control when required. The system tracks the driver's cognitive state and attentiveness, using biometric sensors and behavioral analysis to assess when human intervention is necessary. This dual approach—ethical decision-making and driver engagement monitoring—ensures that the vehicle operates both autonomously and responsibly while maintaining critical human oversight.

4. In-Cabin Threat Detection: Safety extends beyond the vehicle's external environment. MANTHRA-X includes an intelligent, multi-modal in-cabin threat detection system that monitors the internal environment for signs of distress, unauthorized activity, or other potential threats. This system ensures that the vehicle can respond to internal security risks promptly, providing an added layer of protection for both the driver and passengers.

Each of these components plays a crucial role in enhancing the autonomous driving experience. Together, they ensure that the vehicle operates autonomously with a high degree of reliability while safeguarding the safety and well-being of its occupants. The integration of these technologies into a single, cohesive system makes MANTHRA-X a comprehensive solution to the complex problems faced by autonomous vehicles today.

What sets MANTHRA-X apart from other existing AV solutions is its ability to integrate critical systems that address both the external and internal safety challenges of autonomous driving. While many current autonomous vehicle systems focus primarily on aspects like object detection or collision avoidance, MANTHRA-X uniquely brings together these technologies into a unified framework that also addresses ethical decision-making and driver engagement. By doing so, it tackles some of the most pressing concerns in the field, ensuring the vehicle can operate safely and ethically while maintaining a human-centric approach to driving.

The research conducted in this project aims to bridge the gaps in existing technologies by creating a comprehensive and adaptive system that can not only ensure safe and reliable autonomous driving but also provide a more ethically aware, responsive, and user-centered experience. This project also seeks to advance the ongoing research in autonomous vehicle integration by providing real-time decision-making algorithms capable of responding to dynamic environments and complex ethical dilemmas.

This report outlines the development of MANTHRA-X, detailing each of its components, the technologies used, the challenges encountered, and the results from its testing and validation. By focusing on these critical aspects of autonomous driving, MANTHRA-X is positioned to be a significant contributor to the future of autonomous mobility, paving the way for a safer, more reliable, and ethically responsible form of transportation.

1.1 Background Literature

1.1.1 Object detection and motion prediction in autonomous vehicles

The transportation sector is experiencing a transformative shift driven by the rise of autonomous vehicle (AV) technologies. These systems are designed not only to enhance traffic flow but also to improve road safety and minimize environmental impact. As urbanization continues to accelerate globally, challenges such as traffic congestion and road accidents have become increasingly pressing. The incorporation of advanced Artificial Intelligence (AI), computer vision, and machine learning techniques into AV systems represents a groundbreaking approach to addressing these critical issues.

Autonomous vehicles rely heavily on accurate perception and decision-making systems to navigate safely. The core functionalities include object detection, behavior prediction, localization, and path planning. The perception module plays a critical role, as it provides the vehicle with an understanding of its surrounding environment through sensor fusion (Camera, LiDAR, radar, GPS).

Recent progress in deep learning—particularly in Convolutional Neural Networks (CNNs), Graph Neural Networks (GNNs), and Transformer architectures—has significantly shaped the development of perception systems in autonomous vehicles. Object detection models like YOLOv5 have gained prominence for their efficient trade-off between accuracy and real-time performance [1][2]. CNNs are widely used to interpret spatial patterns in images, supporting tasks such as lane detection and steering angle prediction [6]. Meanwhile, GNNs effectively capture the interactions among dynamic agents in traffic environments. When integrated with Transformers, known for their strength in sequence modeling and attention mechanisms [5], GNN-based models can achieve enhanced capabilities in predicting the motion and behavior of nearby vehicles [14].

Numerous studies and projects have leveraged these models within simulation environments like CARLA [3] and real-world AV testbeds to evaluate their effectiveness. However, most existing frameworks treat perception, behavior prediction, and navigation as separate components. Manthra-X introduces an integrated solution that unifies these tasks for better consistency and real-time performance.

1.1.1.1 Historical Overview of Autonomous Vehicles

The concept of autonomous vehicles dates to the mid-20th century, but it was not until the late 2000s that practical advancements began to emerge with the introduction of deep learning and real-time simulation environments. Companies like Waymo, Tesla, and NVIDIA spearheaded research and development initiatives by introducing vehicle prototypes embedded with perception and control systems powered by deep neural networks.

The evolution of perception technologies began with basic computer vision techniques, such as edge detection and histogram analysis, gradually shifting to object classifiers like Haar cascades and Histogram of Oriented Gradients (HOG). These approaches, however, were sensitive to lighting and occlusions and lacked generalizability.

The advancement of Convolutional Neural Networks (CNNs) significantly enhanced object detection and image segmentation tasks. Foundational models such as AlexNet, VGGNet, and ResNet established effective hierarchical feature extraction techniques, which greatly improved image classification in complex urban environments. The YOLO (You Only Look Once) family of models transformed real-time object detection by reimagining it as a unified regression task [1]. YOLOv3 achieved a strong trade-off between speed and accuracy, while subsequent versions like YOLOv4 and YOLOv5 further refined performance by improving backbone architectures, activation functions, and incorporating advanced data augmentation methods [2].

1.1.1.2 State-of-the-Art Object Detection with YOLOv5

YOLOv5 is a PyTorch-implemented object detection model that provides a streamlined training pipeline, enhanced feature pyramid networks, and auto-learning bounding box anchors [1]. Its variants (YOLOv5s, m, l, x) provide trade-offs between computational demand and accuracy. In the context of autonomous vehicles, YOLOv5's real-time detection capabilities make it suitable for embedded platforms.

YOLOv5 has been adopted in traffic detection systems, pedestrian tracking, and vehicle localization. Studies report improved generalization to multiple classes in a single frame while preserving a high inference rate. However, it still suffers in lowlight and occludes environmental problems addressed in part by ensemble approaches or sensor fusion.

1.1.1.3 Lane Detection and Steering Prediction using CNNs

researchers Lane detection is central to AVs for lateral vehicle control. Traditional imageprocessing methods relied on Hough Transforms and polynomial fitting to identify lane boundaries, but they lacked adaptability to different road textures, faded markings, or occlusions. Deep learning introduced CNNs as an efficient alternative, capable of extracting road edges, lane types, and curvature from raw images [6]. End-to-end approaches such as NVIDIA's PilotNet and Tusimple Benchmark winners use CNNs to map input frames directly to steering angles. Transfer learning and LSTM hybrids have also been proposed for memory-augmented driving systems.

This project involves training a CNN-based model using RGB and semantic segmentation images obtained from the CARLA simulator, enabling the extraction of both high-level and finegrained visual features. The primary objective of the model is to predict steering angles, and it is subsequently utilized as a visual feature extractor within a reinforcement learning framework.

1.1.1.4 Behavior Prediction with GNN + Transformer Models

Traditional motion prediction techniques used Kalman Filters and Gaussian Processes. While effective in low-noise environments, these models fail in interaction-rich urban scenarios. LSTM-based trajectory models improved performance by capturing temporal dependencies.

Graph Neural Networks (GNNs) represent traffic agents (vehicles, pedestrians) as nodes and their interactions as edges. GNNs such as GraphSAGE, GAT, and DGCN are used to propagate temporal and spatial information between agents [14].

Transformers further refine this by using multi-head self-attention, allowing for dynamic weighting of past and future interactions [5]. The combination—GNN +

Transformer—has shown superior results in datasets like Argoverse, Waymo Open Dataset, and CARLA simulations.

In this study, a hybrid model with GraphSAGE layers followed by TransformerEncoder blocks processes positional, directional, and velocity embeddings to classify vehicle behavior into Going, Coming, Parked, or Crossing.

1.1.1.5 Integration in Simulation Environments

CARLA Simulator is an open-source AV simulation environment that provides sensor data (camera, LiDAR, radar) in urban and rural scenes [3]. CARLA allows for generating groundtruth data for training models under varied lighting, weather, and traffic conditions.

The Manthra-X system was entirely implemented and tested in CARLA, with synchronized data pipelines across all modules. Model outputs such as bounding boxes, steering commands, and behavior labels were evaluated against simulation trajectories and collision statistics.

1.1.2 Decision Making and Collison Avoidance

1.1.2.1 Deep Reinforcement Learning (DRL)

Deep Reinforcement Learning (DRL) is a powerful fusion of reinforcement learning and deep learning that enables autonomous systems to make decisions in real world complicated environments through trial and error. DRL systems differ from traditional rule based or behavior based models in that they learn optimal control policies by directly interacting with the environment through a feedback signal referred to as a reward. In autonomous vehicle (AV) systems, this allows for more flexibility to track dynamic and unpredictable driving conditions [3][11].

One of the best known DRL algorithms is Deep Q Network (DQN), which synergizes Q learning and deep neural networks to provide an estimate for the value of different actions under a given state. This helps AVs to learn discrete action policies such as braking, changing lanes, or accelerating. Nevertheless, DQN is challenging to apply with continuous action spaces and unstable when applying it to manipulate high dimensional inputs, e.g., camera or LiDAR inputs [4].

To overcome these limitations, policy gradient methods like Proximal Policy Optimization (PPO) are gaining prominence. PPO is an on policy algorithm that has a tradeoff between performance and learning stability through limiting policy improvements with a clipped surrogate objective. It is hence extremely well suited to the class of problems involving continuous control where stable and smooth actions need to be obtained like modulating steering angles or throttle values [9]. PPO is particularly beneficial in real driving situations where sudden, extreme policy changes may lead to dangerous behavior or system instability.

In AV scenarios, PPO is typically trained with good quality reward functions that encourage safe actions and punish hazardous maneuvers. Rewards can be issued for maintaining a safe distance, staying in a lane, or obeying traffic regulations, while penalties are imposed for accidents, tight turns, or abrupt accelerations. The ability to create reward schemes allows DRL models to learn to adapt to various driving behaviors, traffic regulations, and surroundings [5].

However, PPO based systems are poor at long term planning, particularly in cases where actions have delayed rewards. For example, while driving through a busy intersection, the right action might not receive an immediate reward, making it difficult for PPO to assign credit properly. Moreover, PPO requires substantial data and computation, particularly when dealing with high dimensional sensor inputs and large simulated worlds [14].

To address these challenges, researchers have explored the combination of Monte Carlo Tree Search (MCTS) and DRL. MCTS is a heuristic search algorithm that approximates numerous potential future action sequences before making a decision. In hybrid DRL MCTS systems, the DRL model provides a value estimate or policy prior, and MCTS explores multiple possible paths. This strategy improves long term vision and risk estimation, enabling Avs to explore multiple possible future states and opt for actions that reduce the likelihood of collisions. In those situations like merging onto a highway or passing a slow-moving vehicle, MCTS can predict several trajectories and analyze the most secure choice based on potential future interactions with surrounding vehicles or pedestrians [12].

This PPO with MCTS blend works at maximum performance levels at high risks that are known in the course where risk conditional decision-making stands at most superior. On the other hand, hybrid forms will always impose added computational latency along with complications mainly for edge deployments inside real time on any Edge Computing of in vehicle machines.

1.1.2.2 Classical Control Methods and Their Limitations

Before the advent of learning-based approaches, conventional control methods such as Proportional-Integral-Derivative (PID) and Model Predictive Control (MPC) were the backbone of AV control systems. These methods rely on mathematical models and feedback control to determine control actions in real-time [14].

PID controllers are error correct: they continually modify control inputs based on the difference between desired and real system states. Their strength is simplicity, dependability, and quick response, especially in clearly defined, predictable environments such as highways or closed loops. In AVs, PID has been used widely for lane keeping, path following, and adaptive cruise control, where the vehicle must follow a set distance and speed relative to other vehicles [13].

While practical, PID controllers suffer from several limitations. They are non adaptive, in the form that they are unable to adapt their behavior when facing unknown or non-linear situations like sudden pedestrian appearances, hostile drivers, or road blocks. Secondly, PID parameter tuning by hand is time-consuming and cannot be generalized to other driving situations [15]. These limitations render PID unsuitable for collision avoidance in dynamic and uncertain settings such as urban traffic where interactions are complex and unpredictable. Model Predictive Control (MPC) is a better alternative to PID as it uses a predictive model to schedule a sequence of control inputs over a horizon. MPC optimizes a cost function, typically incorporating vehicle dynamics, constraints, and predictions of the future at each time step by solving an optimization problem. Whereas MPC has greater planning capability and constraint handling compared to PID, it is still highly reliant on the model of the system and environment dynamics, which can prove challenging to update in actual application. The computational complexity of MPC also grows with the planning horizon length and model complexity.

1.1.2.3 Hybrid Approaches for Enhanced Collision Avoidance

From comprehending the strength and limitations of classical control methods and DRL methods, researchers proposed hybrid mechanisms that combine the adaptability of DRL with the robustness and readability of conventional controllers. A possible approach is using DRL at high-level choices like path selection or maneuver planning while utilizing classical controllers like PID or MPC at low-level control such as throttle control or steering angle control [7][16].

These hybrid control systems take advantage of DRL's ability to learn adaptive policies from experience without losing the stable and constraint-imposing predictable action of PID or MPC. For example, in a highway merging task, the DRL agent may determine the optimal merging policy by predicting drivers' intentions surrounding it, while the PID controller provides smooth lane change based on the selected path. Another direction of interest is the integration of Graph Neural Networks (GNNs) into DRL pipelines. GNNs are particularly appropriate for AV scenarios where interactions between multiple agents vehicles, pedestrians, cyclists must be considered [6]. By mapping these entities onto nodes in a graph and their interactions onto edges, GNNs enable spatiotemporal reasoning that helps AVs predict the motion and intent of surrounding agents [10]. For example, in an overcrowded urban intersection, DRL enhanced with GNN would enable an AV to estimate whether pedestrians may walk across or where other cars could turn and modify its own path accordingly so that it avoids crashes.

Real-time sensor fusion, combining the information from LiDAR, radar, cameras, and GPS, is also crucial in such applications. Sensor fusion improves the situational awareness of the AV and provides rich inputs to DRL models. Combined with GNNs, this enables context-aware decision-making even in noisy or uncertain environments such as poor weather or low-light conditions [8].

While these hybrid approaches look promising, they pose system integration, scalability, and real-time problems. Safe interaction among learned policies and traditional controllers requires careful design and verification. Furthermore, as AVs move toward higher levels of autonomy, these systems must be robust to edge cases and adversarial scenarios not seen during training. Self-driving car collision avoidance has evolved significantly from the employment of traditional PID control, to advanced DRL methods and hybrids. PPO style DRL algorithms possess good learning ability and dynamic adaptability, while hybrids and extensions like MCTS and GNNs push the boundary in long-horizon planning and situation awareness [10][12]. While computationally demanding, these sophisticated methods have the potential for safer, more effective driving in ever-more challenging real-world driving scenarios. Further research into hybrid control architectures, reward engineering, and sensor integration is critical to realizing complete autonomy with low collision risk.

1.1.3 Ethical Decision-Making and Driver Awareness monitoring in Autonomous Vehicles

1.1.3.1 Ethical Decision-Making

The emergence and continuous advancement of autonomous vehicles (AVs) represent a paradigm shift in modern transportation. As AVs increasingly move toward full autonomy, their capability to manage morally complex scenarios becomes not only a technical challenge but also a profound ethical imperative. These vehicles are required to make instantaneous decisions with significant consequences, sometimes involving trade-offs between competing moral values. Among the most discussed examples is the autonomous version of the "trolley problem"—an ethical dilemma in which an AV must decide whether to prioritize the lives of its passengers or pedestrians in an imminent accident [1].

Conventional AV systems have largely operated using deterministic, rule-based algorithms aimed at maximizing safety and legal compliance. These systems follow hard-coded instructions for braking, turning, or lane changes based on sensor data and environmental analysis [2]. While effective in structured scenarios, such approaches are ill-suited for dynamic environments involving ambiguous moral decisions or conflicting goals, especially in urban settings where pedestrians, cyclists, and other vehicles interact unpredictably.

To resolve this limitation, researchers have sought to integrate classical ethical frameworks into AV decision-making. Utilitarianism focuses on maximizing total benefit or minimizing total harm, even if it means sacrificing the few for the many [3]. Deontology emphasizes adherence to strict rules or duties, such as not harming others intentionally, regardless of the consequences [3]. Virtue ethics encourages behavior aligned with moral character traits like fairness, honesty, and compassion [3]. While conceptually helpful, these frameworks are static and do not easily scale across rapidly changing contexts or cultural expectations.

As a result, machine learning—particularly Reinforcement Learning (RL)—has emerged as a promising solution. RL allows AVs to learn from trial-and-error interactions within simulated environments, refining their decision-making policies based on feedback. Deep Q-Networks (DQN), a type of RL model, utilize neural networks to approximate the value of taking a particular action in a given state, enabling complex decision strategies to evolve over time [4].

Abel et al. [4] demonstrated how RL can serve as a flexible structure for encoding ethical behaviors by mapping outcomes to ethically weighted rewards. This technique permits AVs to learn not just safe driving strategies but morally reasoned ones. Chien et al. [5] expanded on this by training DQN agents in simulated driving environments where ethical trade-offs were encoded directly into the reward structure. Their agents learned to favor decisions that minimized harm while respecting traffic norms and social expectations.

Another important consideration is the cultural relativity of ethics. Greene et al. [6] introduced a tuning parameter, the "k-value," which shifts the AV's decision orientation from egoistic (favoring passengers) to altruistic (favoring others such as pedestrians). This parameter allows system designers or regulators to customize the ethical stance of the AV based on regional values. This approach is further validated by the Moral Machine study by Awad et al. [7], which collected over 40 million decisions from citizens across 233 countries. The findings underscored wide variations in ethical preferences based on culture, gender, religion, and geography, thereby highlighting the importance of ethical adaptability in AVs.

In a step toward real-world deployment, Detjen-Leal et al. [8] proposed adaptive ethical policies where the vehicle refines its decision logic over time through real-world feedback and scenario logging. This mirrors how human ethical reasoning evolves through lived experience and social context, suggesting a need for lifelong learning mechanisms in AVs.

Furthermore, interpretability and trust are essential for public acceptance. Kuutti et al. [15] and Pan et al. [17] argue for transparent decision systems where AVs not only act ethically but can also justify their decisions to users and regulators. This is crucial for building public confidence and establishing accountability in autonomous systems.

In this research, the ethical decision-making engine was developed as a stand-alone component using Deep Q-Networks trained in a high-fidelity simulation environment. Scenarios included obstacle avoidance with conflicting paths, pedestrian prioritization at uncontrolled crossings, and handling sudden interruptions in vehicle trajectory. A reward function was designed to encode ethical principles such as harm minimization, rule compliance, and social preference weighting. Though independent of the driver awareness system, this module sets the stage for integrated, explainable, and context-sensitive ethical behavior in AVs.

1.1.3.2 Eyeball Tracking Systems and Driver Awareness

As AV systems continue to evolve, human oversight remains essential—particularly in semi-autonomous configurations (SAE Levels 2 and 3), where responsibility alternates between the human driver and the automated system. In such setups, the driver's awareness, attention, and emotional state are pivotal for safe operation. Failures in this human-machine transition are among the leading causes of AV disengagement-related incidents. Therefore, real-time monitoring of driver cognition and awareness has become an indispensable safety feature.

Eyeball tracking is a non-invasive, real-time technique that captures a driver's gaze direction, eye closure, and blink rate using video-based computer vision systems. These physiological signals serve as indicators of cognitive load, visual focus, drowsiness, or distraction. Pioneering work by Doshi and Trivedi [9] established a dual-observation framework—capturing both the driver's eye behavior and the visual scene—to infer attentiveness with high accuracy.

More recent innovations, such as the few-shot learning model by Park et al. [10], utilize convolutional neural networks (CNNs) to personalize gaze estimation with minimal calibration. These models perform robustly across drivers with diverse facial features, seating positions, and lighting conditions, making them ideal for real-world deployment.

Fatigue detection has also benefited from multimodal approaches. Chen et al. [11] and Zhao et al. [12] showed that incorporating facial expressions, blink rate variability, and gaze instability greatly enhances fatigue detection accuracy. One widely adopted metric is PERCLOS (Percentage of Eyelid Closure over the Pupil), which effectively correlates with driver drowsiness.

In addition, emotional state monitoring is gaining importance as a safety metric. Wang et al. [13] integrated facial emotion recognition with gaze behavior to identify psychological states like frustration or overconfidence. These emotional cues are vital for adaptive AV systems that seek to personalize interactions or preempt risky behavior.

The Eyeball Tracking System developed in this research uses MediaPipe to extract 468 facial landmarks in real-time. The system computes Eye Aspect Ratio (EAR) to detect prolonged eye closure and maps gaze zones based on iris positioning. The driver's visual focus is classified into three zones:

- Central: Aligned with the road ahead
- Peripheral: Monitoring mirrors or dashboard
- Off-road: Distracted or inattentive behavior

What distinguishes this implementation is the integration of a mobile application for haptic and visual alerting. When distraction or fatigue is detected, the system sends an alert to the mobile app, which displays a notification and simultaneously triggers the phone's vibration motor. This dual-channel feedback—visual and tactile—has been shown to significantly reduce driver reaction time in distraction scenarios.

According to Hossain and Muhammad [16], multi-sensory feedback systems improve safety outcomes and user satisfaction in emotion-aware technologies. Pan et al. [17] support combining gaze, emotion, and contextual information to create holistic and trustworthy human-AI interactions.

The mobile app offers several practical advantages:

- Cross-platform compatibility and ease of deployment
- Reduced reliance on proprietary in-vehicle hardware
- Customizable user interface and threshold settings

Furthermore, the system’s modular architecture supports future enhancements such as:

- Wearable integration for more discrete alerts
- Edge computing for reduced latency and better privacy
- Real-time data fusion with AV control logic to inform automated maneuvers

Although this system currently operates independently of the ethical decision engine, it contributes substantially to AV safety by ensuring human readiness. By combining real-time gaze analytics with intuitive mobile alerts, it strengthens the reliability of shared-control systems.

This chapter presented a comprehensive review of two critical dimensions in autonomous vehicle design: ethical decision-making and driver awareness monitoring. The ethical decision model leverages Deep Q-Networks to simulate moral reasoning in complex scenarios, addressing cultural variability and adaptive learning. Simultaneously, the Eyeball Tracking System employs facial landmark detection and mobile-based feedback to monitor and enhance driver attentiveness. While each module was developed independently, their integration in future systems promises to deliver ethically sound, human-aware AVs that are responsive, transparent, and safe. These foundational systems not only improve vehicle intelligence but also foster trust and acceptance among users and regulators, setting a strong base for the methodology and implementation discussed in the following chapters.

1.1.4 An Intelligent Multi-Modal In-Cabin Threat Detection System for Autonomous Vehicles

Developments to introduce in-cabin security systems has gained considerable attention in recent years, driven by the need to enhance passenger safety and protect against potential threats Merging real-time camera feeds and audio inputs is a significant development in this field, leveraging sophisticated technologies to monitor and analyze the inside environment of vehicles.

Image recognition, especially using convolutional neural networks (CNNs), has become a key element in modern security systems. CNNs are a type of deep learning algorithm that excels at processing visual information. They've been thoroughly researched and are widely used in various areas, such as facial recognition, object detection, and understanding scenes. Because CNNs can identify and classify objects in images with high precision, they are perfect for spotting security threats like weapons and unauthorized items in vehicle cabins. Research indicates that CNNs can be trained on large datasets to recognize specific objects and behaviors, making them a strong foundation for security-related applications (Krizhevsky, Sutskever, & Hinton, 2012).

Besides, the natural language processing (NLP) and machine learning have also made significant improvements in voice recognition technology. It uses audio inputs to identify speakers, transcribe speech and even detect certain words or tones (e.g. sentiment analysis). These systems take in huge amounts of data to train on sounds and noises, which make them capable enough to tell apart legit voices from those that are not. There has been research to evidence the efficacy of voice recognition in security applications, for example recognizing expressions or phrases (Rabiner & Juang 1993). By creating a database of authorized voice patterns, security systems can quickly identify anomalies and respond appropriately.

One of the key capabilities security systems should have been Anomaly detection — identify patterns or behaviors that are different from what we know to be normal. It is also capable of working with both visualization and audio data. Anomaly detection would be used to detect the presence of a weapon or aggressive speech in- cabin at an car. Anomaly detection methods regularly employ machine learning powered by trained models on normal behavior, highlighting differences as suspect menaces. Research has shown that several Apriori techniques work well in security contexts . Studies have demonstrated the effectiveness of these techniques in various security scenarios, enhancing the ability of systems to detect and respond to threats in real time (Chandola, Banerjee, & Kumar, 2009).

The importance of real time processing, in security systems cannot be emphasized enough. Immediate analysis enables detection and response to threats reducing risks for those inside. Progress in computing power and algorithm efficiency has made real time processing allowing security systems to monitor and analyze data continuously from sources. Research has examined architectures and frameworks for real time security systems stressing the need for speed and accuracy in detecting threats (Gadepally et al., 2013).

Past research on in cabin security systems has mainly concentrated on components like image recognition or voice analysis. However there is a rising trend towards integrating technologies to form security solutions. Recent investigations have looked into combining image and voice recognition with monitoring showcasing the potential for robust and precise security systems (Zeng et al., 2019). This integration of technologies marks an advancement in the field offering an approach to, in cabin security.

1.2 Research Gap

The autonomous vehicle (AV) industry has made great strides toward full autonomy, yet Level 3 autonomous vehicles, which require driver oversight and intervention under certain conditions, present significant challenges that remain underexplored in the research. Current studies have predominantly focused on autonomous driving at higher levels of automation, such as Levels 4 and 5, where no human intervention is required. However, Level 3 vehicles, those that can operate autonomously within specific limits, still require robust support systems to assist the driver. Existing systems largely focus on automating driving tasks but fall short when it comes to providing sufficient driver assistance, real-time decision-making, and ethical frameworks for situations where the driver must still engage with the system. The MANTHRA-X support system integrates multiple systems aimed at improving the vehicle's safety, responsiveness, ethical decision-making, and driver engagement. While various technologies such as object detection, motion prediction, and collision avoidance have been studied extensively, the integration of these technologies with real-time adaptive decision-making and driver awareness remains a major gap. Current solutions either focus on automated driving or driver monitoring, but few systems successfully combine the two in a way that accounts for the driver's mental and physical state and adapts to unpredictable real-world driving conditions.

One of the most significant gaps in research lies in the lack of integrated systems that combine different autonomous driving functions into a unified support system for Level 3 AVs. Research like Bojarski et al. (2016) and Cui et al. (2016) shows that individual subsystems such as object detection, collision avoidance, and path planning can be highly effective when used separately, but the integration across these components to ensure real-time adaptive decision-making and driver assistance is still lacking. Existing systems fail to provide the real-time context-aware decision-making needed to make vehicles both autonomous and responsive to driver behavior and environmental changes. MANTHRA-X addresses this by combining object detection, motion prediction, collision avoidance, and ethical decision-making into a single cohesive support system. This integration ensures that the vehicle can detect and predict objects, make context-aware decisions in real-time, and assist the driver in cases of fatigue or distraction, intervening when necessary.

Ethical decision-making remains one of the most controversial aspects of autonomous vehicle research. Current systems are often designed to handle simple or predictable scenarios, but when faced with unavoidable moral dilemmas—like deciding whom to harm in a crash scenario—existing systems tend to fall short. The trolley problem, often cited in philosophical debates, is a useful theoretical tool for examining these issues, but real-world applications of these ethical models are far more complex. Existing frameworks are mostly theoretical, with limited practical implementation in real-time decision-making systems. Lin (2016) and Goodall (2014) have explored these concepts, but they do not offer practical solutions for Level 3 AVs that require dynamic ethical decision-making in emergency situations. MANTHRA-X overcomes this challenge by integrating an ethical decision-making framework that uses principles of utilitarianism and deontological ethics, allowing the vehicle to make decisions based on real-time data and societal norms. This hybrid approach ensures that the vehicle can handle complex ethical dilemmas and make decisions that reflect moral considerations in real-world driving conditions.

Another critical gap exists in the area of driver awareness and cognitive state monitoring. Unlike fully autonomous systems, Level 3 AVs still require the driver to be present and ready to take over control in case of a system failure. While traditional systems monitor basic metrics such as eye movement or facial recognition, they are often insufficient for assessing cognitive fatigue, disengagement, or mental readiness to intervene. Existing studies, such as those by Zhang et al. (2017) and Liu et al. (2018), highlight the limitations of these systems, as they fail to detect subtle signs of fatigue or distraction that could delay driver intervention in an emergency. MANTHRA-X addresses this gap by incorporating biometric sensors, neurocognitive monitoring, and AI-driven behavioral analysis to continuously assess the driver's cognitive state. This multi-layered approach ensures that the system can accurately detect driver fatigue, distraction, and mental disengagement, and it provides real-time feedback or automatically takes control when needed. This real-time monitoring significantly improves safety, ensuring that the driver remains alert and capable of reacting in critical situations.

There is also a gap in in-cabin threat detection, which has largely been overlooked in autonomous vehicle research. While significant progress has been made in detecting external threats such as pedestrians and vehicles, the driver's health, mental state, and potential in-cabin threats have received little attention. Level 3 AVs need to ensure the safety of the driver and occupants, especially in the event of health emergencies, psychological distress, or unauthorized access to the cabin. Amarasinghe et al. (2020) and Yuan et al. (2018) have explored some aspects of driver behavior monitoring, but they fall short in providing solutions for real-time threat detection in the cabin that accounts for health emergencies and driver distress. MANTHRA-X fills this gap by implementing a multi-modal in-cabin monitoring system, integrating cameras, microphones, and biometric sensors to detect signs of driver distress and health emergencies. This system also monitors for unauthorized access to the cabin, providing a comprehensive safety net for driver protection.

Finally, there remains a research gap in the real-time adaptive decision-making of autonomous systems, especially in highly dynamic environments. Many current AV systems rely on pre-programmed algorithms that can't respond quickly enough to sudden changes in traffic conditions, driver behavior, or obstacles. These systems may struggle to handle the complexity and unpredictability of real-world driving. Existing research on reinforcement learning (as seen in Silver et al. (2016)) has made strides in autonomous decision-making, but many systems are limited to isolated subsystems like path planning or collision avoidance [6]. MANTHRA-X overcomes this limitation by using sensor fusion, machine learning, and real-time data processing to enable the system to adapt to changing driving conditions. This continuous integration of data ensures that MANTHRA-X is capable of handling dynamic driving environments with minimal delay, improving safety and responsiveness.

The research presented here contributes significantly to the development of Level 3 AVs by addressing the gaps in system integration, ethical decision-making, driver engagement, real-time adaptability, and in-cabin threat detection. MANTHRA-X offers a holistic approach that integrates all these components into a single, comprehensive support system that ensures safe, responsive, and ethically sound operation of autonomous vehicles.

Table 1 Comparison Between Existing Systems and Proposed System

Features	**[1]**	**[5]**	**[6]**	**[7]**	**[8]**	**[9]**	**MANTHRA-X**
Integrated Object Detection and Motion Prediction	Yes	Yes	No	No	No	No	**Yes**
Real-Time Collision Avoidance with Sensor Fusion	Yes	Yes	No	No	No	No	**Yes**
Real-Time Adaptive Ethical Decision-Making Framework	No	No	No	No	Yes	Yes	**Yes**
In-Cabin Threat Detection	No	No	No	No	No	No	**Yes**
K-Value-Based Ethical Prioritization	No	No	Yes	No	No	No	**Yes**
Adaptability to Complex Driving Scenarios (e.g., traffic, obstacles)	Limited	Limited	Yes	Limited	Yes	Yes	**Yes**
Driver Mobile Alert and Assistance System	No	No	No	No	Yes	No	**Yes**

1.3 Research Problem

1.3.1 Object detection and motion prediction in autonomous vehicles

Autonomous vehicles struggle to perceive and react effectively in complex urban environments due to the fragmented nature of current perception systems, which separately handle object detection, lane following, and motion prediction. This siloed approach leads to inefficiencies, higher latency, and poor agent interaction modeling. The study aims to address this by developing a unified, real-time perception framework that integrates YOLOv5 for object detection, CNNs for lane discipline, and GNN-Transformer models for behavior-aware motion prediction. The goal is to enhance contextual understanding, reduce latency, and improve AV performance in dynamic, multi-agent traffic scenarios using the CARLA simulator.

1.3.2 Autonomous vehicle collision avoidance

Despite all the advancements in Autonomous Vehicle (AV) technologies, reliable and effective decision-making for collision avoidance remains a fundamental issue for real-world deployments. With the exception of a few recent Deep Reinforcement Learning (DRL) systems, although promising in simulation, no current DRL systems can generalize to real-world situations owing to unexpected phenomena such as sensor noise, non-stationary human behavior, and rare edge processing case occurrences. Besides, current DRL models also neglect adherence to traffic regulations and ethics in unavoidable crashes, raising concerns about safety, legality, and social acceptability. Technically, the systems also have the problem of high computational needs, posing difficulties to real-time deployment on embedded vehicle platforms. Furthermore, current methods also lack robust mechanisms to deal with complex multi-agent interactions and long-term trajectory planning, which are problems of concern in congested, dynamic traffic environments. Resolving these issues is required so that autonomous cars can safely, accurately, and contextually make decisions in real-time and also be feasible enough to be put on various vehicle platforms.

1.3.3 Ethical decision making and driver awareness monitoring in autonomous vehicles

Autonomous vehicles (AVs) face significant challenges in making ethically sound decisions in complex, real-world scenarios. While Deep Q-Networks (DQN) offer potential for learning ethical behavior through simulation, existing models often neglect human-centric factors such as driver awareness, emotional state, and cultural values. Simultaneously, driver monitoring technologies (e.g., gaze tracking, fatigue detection) operate independently without influencing ethical decision-making processes. Current systems also lack integration with dynamic ethical prioritization (e.g., k-value tuning) and multisensory feedback mechanisms. This research addresses the critical gap by proposing a unified framework that integrates ethical decision-making via DQN with real-time driver monitoring, adaptive ethical tuning, and mobile-based feedback to enhance AV safety, trust, and accountability.

1.3.4 An Intelligent Multi-Modal In-Cabin Threat Detection System for Autonomous Vehicles

As autonomous vehicles (AVs), particularly Level 3 models, reduce the need for constant driver oversight, in-cabin security becomes increasingly critical. Existing systems lack real-time, intelligent threat detection that accounts for behavioral and emotional context. This research aims to develop a multimodal AI-based in-cabin security system that integrates video, audio, and emotional data to detect threats such as unauthorized object possession, violent speech, or distress signals under challenging conditions (e.g., low light, background noise). Key challenges include ensuring real-time responsiveness, fusing multimodal data effectively, minimizing false alerts, and maintaining passenger privacy. The system is designed to operate on edge devices, ensuring quick detection and timely alerts to enhance passenger safety in modern AVs.

1.4 Research Objective

1.4.1 Main Objective

The MANTHRA-X project sets out to develop an innovative and comprehensive support system for autonomous vehicles (AVs), aiming to elevate the interaction between the vehicle's autonomous capabilities and the driver's needs in a way that enhances safety, reliability, ethical decision-making, and overall user experience. The project seeks to address the challenges associated with integrating autonomous driving technologies with real-time driver assistance systems that can adapt to complex, unpredictable driving conditions. It focuses on creating a holistic framework that ensures the vehicle can function autonomously when possible, while simultaneously providing the driver with comprehensive support in scenarios that require human intervention.

At its core, the main objective of the MANTHRA-X project is to bridge the gap between the vehicle's autonomous functionality and the driver's cognitive and physical readiness by developing an advanced support system that operates dynamically, responsively, and ethically. This system aims to create a seamless, intelligent interaction between the vehicle and the driver, ensuring that while the vehicle can operate autonomously, it remains deeply integrated with the human driver. The vehicle must recognize when human intervention is required, monitor the driver's state of readiness, and alert or assist the driver accordingly, all while respecting the ethical implications of each decision it makes.

One of the primary challenges that MANTHRA-X addresses is the lack of effective integration between the various subsystems of autonomous vehicles—such as object detection, motion prediction, collision avoidance, and driver monitoring. Existing systems often operate in silos, optimizing each function individually but failing to create a cohesive, real-time support system that operates seamlessly across all these dimensions. MANTHRA-X aims to overcome this by creating an integrated support system that combines sensor fusion, machine learning, and real-time data processing to ensure that the vehicle can make dynamic decisions, adapt to its environment, and respond effectively to both external and internal factors. The system will use multi-layered data—including driver physiological states, environmental conditions, and vehicle status—to make context-aware decisions that keep both the vehicle and the driver safe.

The vehicle's ability to function autonomously in specific scenarios—while also understanding when it's necessary to involve the driver—is another critical aspect that MANTHRA-X seeks to enhance. The support system will continuously assess both the driver's mental readiness and environmental conditions, ensuring that the vehicle can autonomously make decisions when the driver is engaged and can safely request intervention when the driver's focus, attention, or cognitive state deteriorates. This includes real-time monitoring of driver fatigue, distraction, or emergency situations, allowing the system to step in and either alert the driver or take control of the vehicle if necessary.

Another key objective of MANTHRA-X is to develop a real-time ethical decision-making framework that enables the vehicle to handle complex moral dilemmas. Many of today's autonomous vehicle systems lack the ability to make ethically sound decisions in scenarios where harm may be unavoidable—such as during a collision situation where the vehicle must choose between two harmful outcomes. MANTHRA-X aims to create a system that considers both utilitarian principles (maximizing overall well-being) and deontological ethics (adherence to moral duties and rules) when making decisions, ensuring that these decisions align with both societal norms and legal standards. The framework will dynamically adapt based on the situation and real-time feedback, ensuring that the vehicle's actions are as ethically responsible as they are functionally efficient.

In addition to ethical decision-making, the project emphasizes the importance of driver safety and well-being within the vehicle. MANTHRA-X will incorporate in-cabin threat detection to ensure that the driver's safety is not only maintained through external factors but also by monitoring the internal environment for health emergencies, psychological distress, or unauthorized access. The system will use biometric sensors (e.g., heart rate, eye movement, skin conductivity) and emotion detection algorithms to assess the driver's physiological and emotional state. If any abnormalities are detected, the system will respond by alerting the driver or initiating necessary actions, such as slowing down, alerting emergency services, or even taking over the driving tasks until the situation stabilizes.

Furthermore, the MANTHRA-X system is designed to operate in highly dynamic environments—environments that are unpredictable and filled with complex challenges. In real-world driving conditions, the vehicle must be able to react to sudden road obstructions, changes in traffic, or unpredictable behaviors from other road users. Most existing systems rely on pre-programmed rules or static decision-making algorithms that do not fully account for the wide range of possible situations encountered during everyday driving. MANTHRA-X, however, uses machine learning to continuously adapt and improve its decision-making based on real-world data and real-time conditions. This ensures that the system remains responsive and effective no matter how the driving conditions change.

In essence, the main objective of the MANTHRA-X project is to create a comprehensive support system for autonomous vehicles that seamlessly integrates autonomous driving capabilities with driver assistance systems. It aims to ensure that the vehicle can operate autonomously while still being sensitive to the human element—adapting to the driver's state and responding dynamically to the environment. The ultimate goal of MANTHRA-X is to enhance the safety, efficiency, and ethical responsibility of autonomous vehicles, while also ensuring that they are capable of fostering a safer, more engaged, and more responsive relationship between humans and technology on the road.

1.4.2 Specific Objectives

1.4.2.1 Object detection and motion prediction in autonomous vehicles

To develop and train a YOLOv5-based object detection model, tailored specifically for autonomous vehicle scenarios. This involves collecting and annotating traffic data in CARLA, optimizing the model for rapid inference, and validating its accuracy under varied visibility, lighting, and traffic conditions. To design and implement a CNN-based lane-keeping model capable of interpreting road curvature, lane boundaries, and contextual lane features using RGB and semantic segmentation images. The model will be trained to predict steering angles dynamically, enhancing control decisions in real-time. To construct a hybrid GNN + Transformer architecture for behavior classification and motion prediction. This system must effectively represent and learn interactions among dynamic road agents to predict behaviors such as "crossing," "approaching," or "parked." Emphasis is placed on temporal consistency, multi-agent coordination, and generalizability across traffic scenarios. To build an integrated perception pipeline where these three models work cohesively. This includes engineering an interface layer to ensure each module's outputs are synchronized and used to influence downstream planning and control. 9 To evaluate the pipeline performance quantitatively, assessing detection accuracy (mAP), steering prediction error (RMSE), and behavior classification metrics (precision, recall, F1 score). The latency of each module and the entire system will also be benchmarked to determine real-time suitability. To examine the perception system’s interoperability with decision-making modules such as route planning, collision avoidance, and vehicle control systems. This will demonstrate how enhanced perception can lead to safer navigation and adaptive decision-making.

1.4.2.2 Autonomous vehicle collision avoidance

Design and implement a hybrid DQN-PID control mechanism

Combine Deep Q-Network (DQN) for high-level collision prevention with Proportional-Integral-Derivative (PID) control for low-level actuation. This hybrid setup aims to ensure stable and adaptive vehicle control in diverse driving environments.

Improve real-world applicability through robust simulation and transfer learning
Train the model in the CARLA simulator with variable conditions and sensor noise, then explore techniques like domain adaptation and adversarial scenario testing to improve generalization to real-world conditions.

1.4.2.3 Ethical decision making and driver awareness monitoring in autonomous vehicles

Eyeball Tracking for Driver Attention Monitoring:
This objective develops a real-time, non-intrusive driver attention monitoring system using a standard camera and MediaPipe. It analyzes facial landmarks, eye movements, blink rate, and gaze direction to classify the driver's focus (on-road, peripheral, or distracted). The system aims to enhance safety and could be integrated with ethical decision-making models in the future.
Ethical Decision-Making via Deep Q-Networks (DQN):
This goal introduces a reinforcement learning approach to train autonomous vehicles in handling ethical dilemmas. The model uses simulated scenarios to optimize a reward function based on ethical principles like harm reduction and rule adherence. The DQN framework is adaptable and lays the groundwork for incorporating culturally sensitive ethical preferences.
Mobile App with IoT-Based Driver Alerts:
This objective designs a mobile app that receives real-time alerts from the attention monitoring system. It uses visual and vibration-based feedback to alert distracted or drowsy drivers. The app enhances situational awareness, supports varied environmental conditions, and is extendable to IoT-connected devices or infotainment systems.
Ethical Transparency and Driver-Aware Personalization:
This part focuses on making ethical decision-making in AVs explainable and customizable. By using a k-value system, the AV can adjust its ethical behavior based on user, legal, or cultural values. The system is designed to be transparent and responsive to the driver's current state, ensuring safer and more socially acceptable AV decisions.

1.4.2.4 An Intelligent Multi-Modal In-Cabin Threat Detection System for Autonomous Vehicles

Fine-Tune Lightweight Models for Inference on Edge Devices

Focuses on adapting lightweight models like MobileNetV2 and YOLOv8s for real-time inference on edge devices by applying techniques such as quantization and pruning. This ensures low-latency performance without compromising accuracy.

Augment Training Data to Simulate Realistic In-Cabin Scenarios

Aims to create robust training data through image/audio augmentation techniques that simulate challenging in-cabin conditions like low light, noise, occlusion, and varied emotions. This improves generalization and domain adaptation.

Build a Modular Data Fusion and Decision-Making Framework

Designs a system that fuses object, audio, and emotion data using weighted confidence scores and correlation logic to trigger alerts only when threat thresholds are met. This balances detection sensitivity and reliability.

Design an Intuitive User Interface for Alert and Control Functions

Creates a dashboard interface to display real-time threat alerts, configure emergency contacts, and adjust sensitivity settings. This makes the system accessible and actionable for different types of users, including fleet managers and rideshare operators.

Create Evaluation Protocols for Threat Simulation and Testing

Implements controlled simulations of various threats and uses evaluation metrics like confusion matrices and ROC-AUC to measure performance and resilience under stress, ensuring the system is reliable and scientifically validated.

2 METHODOLOGY

2.1 Object detection and motion prediction in autonomous vehicles

The methodology is centered around integrating deep learning modules that work collaboratively to enable accurate, context-aware decision-making. Each model—object detection, lane keeping, and motion prediction—was developed and evaluated independently before being integrated into the CARLA simulator to ensure robustness and real-time performance in dynamic traffic scenarios.

YOLOv5s for Object Detection

The first component employs YOLOv5s for real-time detection of essential road entities such as vehicles, pedestrians, traffic lights, and signs. Chosen for its speed and efficiency, YOLOv5s was trained using a custom CARLA-based dataset annotated via Roboflow. Data augmentation techniques (brightness adjustment, blurring, rotation) were applied to improve generalization. Transfer learning with COCO-pretrained weights and fine-tuning across 100 epochs enhanced model performance. The system achieved high accuracy with an inference speed of 45–60 FPS on NVIDIA GPUs.

CNN for Lane Keeping

The second model handles lane keeping via a CNN that predicts the vehicle's steering angle from input images. The CNN architecture includes convolutional layers with ReLU activations, max-pooling, batch normalization, and dense layers. Training data was generated from CARLA, where each image was labeled with a corresponding steering angle. The dataset was split 80:20 for training and validation. Evaluated using MSE and R^2, the model showed high accuracy in predicting steering angles—even under varied lighting and partial lane visibility—proving its effectiveness for lateral control.

GNN + Transformer for Motion Prediction

The final component uses a hybrid architecture combining Graph Neural Networks (GraphSAGE) and Transformers to predict future behaviors of nearby dynamic agents. Traffic scenes are modeled as temporal graphs, with nodes representing agents and edges capturing spatial interactions. The Transformer encoder captures temporal dependencies through self-attention. Each agent is characterized by spatial and motion features like position, velocity, and heading. Trained on over 500 CARLA scenes, the model classifies agent behavior into four categories: Going, Coming, Crossing, and Parked. Performance was evaluated using F1-score, precision, recall, and confusion matrix, showing superior accuracy and generalizability over standalone models.

2.2 Decision Making and Collison Avoidance

The research approach is aimed at developing a hybrid control framework for decision-making and collision avoidance for autonomous vehicles. The system is developed and tested with the CARLA simulator, which offers high-fidelity highway and urban environments, and multi-sensor input support such as RGB cameras, LiDAR, and GPS.

Essentially, the system uses a Deep Q-Network (DQN) for making high-level decisions, and it is trained via reward-based learning to construct optimal navigation policies. The reward function gives a reward for secure actions such as speed and lane-keeping and a penalty for collision and unsafe distance to vehicles. DQN generates discrete driving actions such as braking, lane change, or acceleration based on real-world environment inputs.

In order to have proper implementation of such decisions, the low-level control of a car is performed through a Proportional-Integral-Derivative (PID) controller. A classic controller steers, throttle, and brake actions by handling deviations between desired and actual conditions, resulting in stable and agile actuation.

Both of these control methods are incorporated into a hybrid DQN-PID model, where the DQN selects strategic actions and the PID controller ensures real-time response. Another collision avoidance module is run concurrently, monitoring projected paths and distance from obstacles. When a potential danger is detected, this module overrides the action taken by the DQN for safety.

Besides, an in-real-time alerting system is deployed through a mobile app developed using Flutter. The application is supplied with telemetry data from the AV system and provides in-real-time reports on vehicle operation. It alerts with immediate collisions when a threat is identified, enhancing transparency, monitoring, and operator response.

In order to facilitate real-time execution on embedded platforms, the system is optimized through techniques like model pruning and quantization. Performance is validated across different simulated environments and scenarios to ensure that the solution remains reliable, responsive, and scalable for deployment in the real world.

2.3 Ethical decision making and driver awareness monitoring in autonomous vehicles

The methodological approach adopted in this research is structured around the development of a modular, human-centric, and ethically responsive framework for autonomous vehicles (AVs). The system has been designed with a strong emphasis on simulation-based learning, real-time behavioral monitoring, and cross-platform interaction between machine and human agents. This approach enables the integration of ethical decision-making capabilities with driver state awareness to support safer and context-aware AV operation.

The methodology is divided into three core modules:

1. Ethical Decision-Making via Deep Q-Networks (DQN):

A reinforcement learning model is trained in a simulated environment to handle morally complex traffic scenarios. It uses harm minimization, legal compliance, and customizable ethical preferences (via k-value tuning) to guide decision-making.

2. Real-Time Driver Awareness Monitoring:

Uses computer vision techniques with MediaPipe to track eye movements, blink rate, and gaze direction for detecting drowsiness or distraction. The system runs on Raspberry Pi using a multithreaded backend for efficient real-time processing.

3. Mobile Feedback and Alert System:

A Flutter-based app connected to Firebase delivers immediate visual and haptic alerts to drivers during unsafe behaviors. It also escalates repeated warnings via SMS, ensuring timely intervention and improved driver-AV interaction.

The framework is modular and scalable, allowing each component to function independently or as part of an integrated system. Simulation-based training ensures safe and iterative model development, while IoT-based communication enhances real-world usability and accessibility. Overall, the methodology combines ethical AI, behavioral sensing, and cross-platform communication to create a context-aware, explainable, and adaptable AV support system.

2.4 An Intelligent Multi-Modal In-Cabin Threat Detection System for Autonomous Vehicles

This research follows a structured engineering methodology to build a real-time, multi-modal in-cabin threat detection system for Level 3 autonomous vehicles. The system—named MANTHRA-X—combines image processing, voice recognition, and emotion analysis within a modular architecture, enabling synchronized threat detection and response through five key stages.

Data Collection and Sensor Integration

In the first phase, data is collected from in-cabin cameras and directional microphones placed strategically inside the vehicle. Visual data captures faces, gestures, and objects; audio data includes conversations and environmental sounds. All data is timestamped and synchronized using a shared system clock, stored locally in a structured format, and synced with the cloud to support multi-modal event correlation.

Visual Threat Detection with YOLOv5

This module utilizes YOLOv5 for object detection of potential threats like weapons, smoke, and beverages. The model is trained on a custom dataset with normal and low-light imagery. Preprocessing includes histogram equalization, denoising, and resizing. Inference is performed locally on edge devices to ensure fast, real-time results with high object detection accuracy.

Audio-Based Threat Recognition

Audio analysis focuses on detecting aggressive speech, critical sounds (e.g., gunshots, screams), and unauthorized voice presence. Preprocessed audio features (MFCCs, RMS Energy, etc.) are fed into a 1D CNN classifier. The model achieves 97.82% accuracy after training on an augmented dataset. Source separation techniques like VoiceFilter are used to manage overlapping voices for precise threat identification.

Facial Emotion Recognition

Facial emotions are analyzed using MobileNetV2 through transfer learning. The system is trained on the FER2013 dataset and recognizes seven key emotions. Preprocessing involves converting grayscale to RGB, data augmentation, and class rebalancing. The model achieves 83.23% training and 70.21% validation accuracy. Detected emotional states are cross-referenced with audio and visual cues to enhance psychological threat detection.

Real-Time Integration and Alerts

A central integration engine fuses outputs from all modules to assess threat levels using rule-based logic and weighted probabilities. High-risk combinations (e.g., weapon + scream) trigger SMS alerts, in-cabin alarms, and data logging. The system runs on low-latency edge platforms like Jetson Nano or Raspberry Pi to ensure quick response times.

System Evaluation and Testing

The system is tested in both simulated and real-world scenarios under diverse lighting, passenger conditions, and behaviors. Evaluation metrics include precision, recall, F1-score, system latency, and false positive/negative rates. Iterative testing supports continuous refinement of model thresholds and improves overall detection reliability.

2.4.1 System Architecture

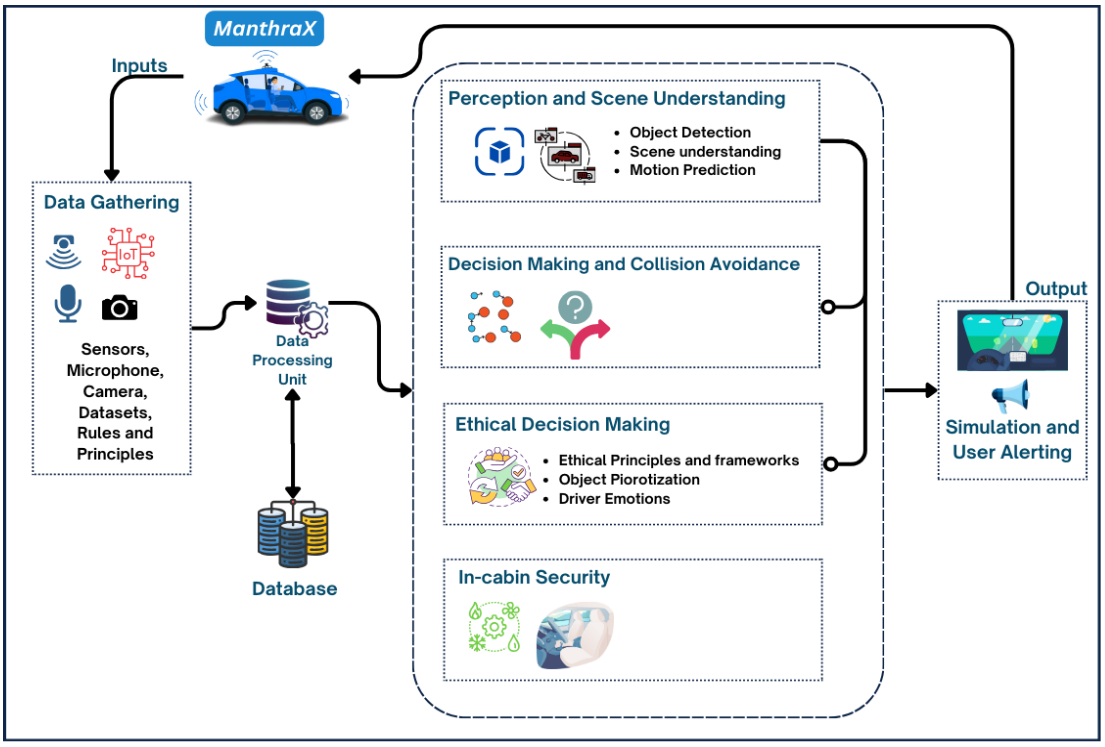


Figure 1 System Architecture

The MANTHRA-X system architecture represents a modular and intelligent framework designed to support vehicle operations by integrating perception, decision-making, and ethical reasoning. The architecture starts with an input layer focused on data gathering through various sensors such as cameras, LiDAR, radar, microphones, and GPS, along with rules and datasets. This raw input is transmitted to a central data processing unit, which integrates real-time sensor data with historical records from a connected database to enable efficient analysis. The processed data feeds into the Perception and Scene Understanding module, responsible for object detection, motion prediction, and interpreting the surrounding environment. Based on this understanding, the Decision Making and Collision Avoidance module applies deep reinforcement learning algorithms such as DQN, combined with classical control and PID controller, safe navigation maneuvers under dynamic driving conditions. Complementing this is the Ethical Decision-Making module, which incorporates ethical principles, object prioritization, and driver emotion analysis to support human-centric and socially acceptable actions. In-cabin security further ensures occupant monitoring and internal safety through anomaly detection and behavior analysis. Finally, all outputs are channeled into a Simulation and User Alerting system, which provides real-time updates and warnings through visual displays and audio cues. This architecture emphasizes safety, adaptability, and ethical responsibility, enabling robust and intelligent autonomous driving in complex environments.

2.5 Commercialization of the Product

MANTHRA-X represents a major innovation in the autonomous vehicle sector, offering an integrated support system that goes beyond basic automation. It enhances safety, ethical decision-making, and driver interaction—bridging the gap between fully autonomous systems and human oversight. As the autonomous vehicle market grows, there's a clear need for solutions like MANTHRA-X that promote responsible, safe, and intelligent vehicle behavior.

The market for MANTHRA-X is rapidly expanding. With the increasing adoption of autonomous vehicles, manufacturers and consumers alike are looking for technologies that not only drive but also think and respond ethically and safely. MANTHRA-X addresses this demand by delivering real-time driver monitoring, adaptive decision-making, and collision avoidance—making it highly relevant in the development of advanced driver-assistance systems and next-generation vehicle platforms.

MANTHRA-X has a broad customer base that includes automotive manufacturers, technology firms, fleet operators, and insurance companies. Automakers can integrate the system into vehicles to enable smooth transitions between manual and autonomous control. Technology companies building autonomous systems can enhance their products with MANTHRA-X's real-time decision-making and driver engagement features. Fleet operators, especially in the shared mobility space, benefit from increased reliability and safety. Insurance providers also stand to gain by using the data MANTHRA-X generates to better assess risk, promote safer driving, and offer more accurate premium structures.

The commercialization strategy for MANTHRA-X is built around multiple revenue models. A licensing model allows automakers and tech firms to pay for system integration and ongoing updates. This provides a consistent and scalable revenue stream. Additionally, a subscription-based model targets fleet operators and mobility service providers, offering continuous monitoring, analytics, and system upgrades. Another strong revenue stream is based on the system’s data collection capabilities. By offering Data-as-a-Service, MANTHRA-X enables insurers, regulators, and manufacturers to access valuable insights about driver behavior, vehicle performance, and ethical decision-making.

Beyond generating revenue, the adoption of MANTHRA-X brings strategic advantages. It enables companies to stay competitive in a rapidly evolving market by offering enhanced safety and ethical capabilities. As regulatory requirements tighten and consumer expectations grow, MANTHRA-X helps organizations stay compliant while also differentiating their products through a focus on responsible autonomy.

Insurance is a key area where MANTHRA-X adds value. Autonomous vehicles introduce new risks related to accidents, system failures, and ethical challenges. By providing real-time risk monitoring and adaptive responses, MANTHRA-X can help reduce accidents and liability, which in turn supports lower insurance premiums. This makes it appealing not only to insurers but also to fleet operators and consumers seeking safer, more cost-effective solutions.

Consumer trust in autonomous technology is crucial, and it often hinges on how vehicles handle complex or ethical decisions. MANTHRA-X builds trust by enabling vehicles to act responsibly in critical situations and by monitoring driver readiness to intervene when needed. This transparency and reliability strengthen public confidence in autonomous vehicles and promote broader adoption.

In conclusion, MANTHRA-X offers a powerful commercial opportunity by combining advanced safety features, real-time decision-making, and ethical driver engagement. As demand for autonomous and intelligent vehicle systems increases, MANTHRA-X stands out as a vital tool for manufacturers, fleet operators, and insurers. With its flexible business models and forward-thinking capabilities, MANTHRA-X is positioned to play a key role in shaping the future of autonomous transportation.

2.6 Testing and Implementation

2.6.1 Implementation

2.6.1.1 Object detection and motion prediction in autonomous vehicles

This section provides a detailed account of the implementation, training, and evaluation processes of the perception system. The development stages, including data collection, preprocessing, model training, and validation, were executed in accordance with the previously outlined system architecture. The testing procedures were designed to evaluate each subcomponent independently and then collectively as part of the integrated framework. All experiments and tests were conducted within the CARLA simulator environment to ensure high realism, safety, and repeatability.

Dataset Collection and Preprocessing

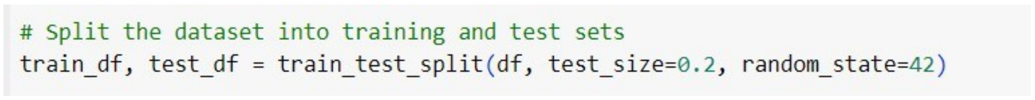
The dataset used for training the object detection, lane keeping, and motion prediction models was generated entirely using the CARLA simulation platform. CARLA offers an ideal testbed for collecting high-fidelity driving data, thanks to its support for configurable cameras, LiDARs, and semantic labeling features.

For the object detection task, the RGB camera was mounted on the ego vehicle and configured to simulate a front-facing dash camera. A custom environment was created that included multiple dynamic actors such as pedestrians, other vehicles, and traffic signs. Images were captured in varying lighting and weather conditions (sunny, rainy, dusk, etc.) to ensure model robustness.

Bounding box annotations for each object class were created using Roboflow's annotation interface and exported in YOLO format.

In the case of lane keeping, images were collected using both RGB and semantic segmentation cameras. A continuous stream of images was paired with real-time steering angle data extracted from the vehicle's control system. To enhance the model's generalization capability, various data augmentation methods were utilized, such as adjusting image brightness, introducing Gaussian noise, and applying random horizontal flipping. The steering angle was encoded in the filename

For motion prediction, spatiotemporal data was collected from CARLA by tracking the location, velocity, and orientation of all dynamic actors in the environment. A script was used to log these attributes across time steps, and they were used to build temporal graphs for each scene. Each node in the graph represented an agent, and edges represented proximity-based interactions.
All datasets were normalized, resized (to 320x180 or 416x416 depending on the model), and split into training (80%), validation (10%), and test (10%) subsets to maintain experimental consistency.



```
# Split the dataset into training and test sets
train_df, test_df = train_test_split(df, test_size=0.2, random_state=42)
```

Figure 2 Splitting the dataset

Component Implementation

The implementation phase of the Manthra-X perception system was conducted in three main stages, each corresponding to one of the core deep learning components: the YOLOv5 object detector, the CNN-based lane keeping model, and the GNN + Transformer motion prediction module. Each model was implemented using Python and relevant deep learning frameworks, including PyTorch and TensorFlow, and was tested within the CARLA simulation environment. For the YOLOv5 object detection module, the implementation began by cloning the official YOLOv5 GitHub repository. The architecture used was YOLOv5s (small), suitable for real-time inference due to its low latency. After configuring the model for the custom dataset format, training was executed over 100 epochs using a batch size of 16 and an image size of 416x416. Loss convergence was monitored using mAP@0.5 as the primary metric. Model checkpoints were saved periodically, and the best-performing model was selected based on validation accuracy.

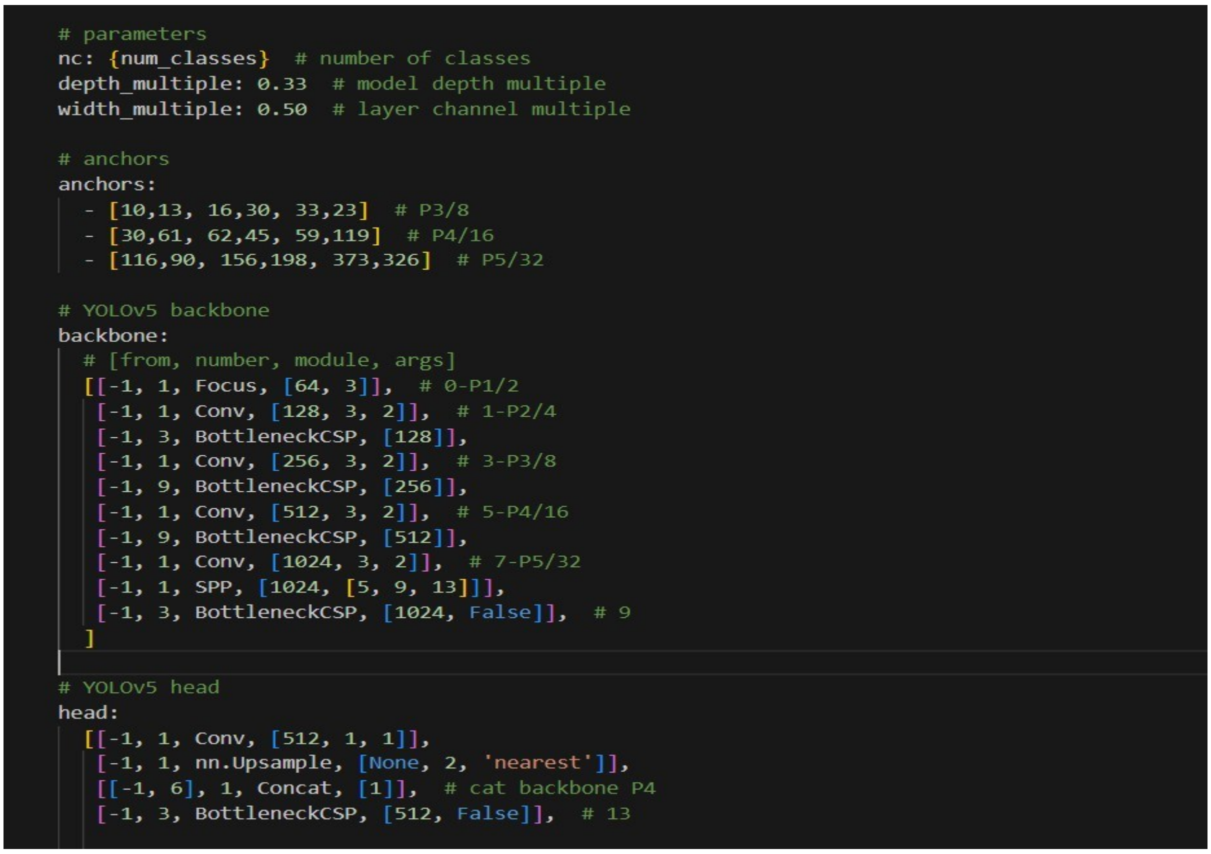
The CNN-based lane keeping model was developed using Keras with TensorFlow as the backend. The model architecture included 3 convolutional layers with ReLU activations, max pooling and dropout layers to prevent overfitting. Dense layers followed the feature extraction stages, ending with a single neuron that outputs the predicted steering angle. Training data was fed from preprocessed image-steering pairs stored in CSV format. The trained model demonstrated consistent performance across a range of road geometries and lighting conditions.

The GNN + Transformer model was constructed using custom Tensorflow modules. Initially, data from CARLA was processed into temporal graph structures using a custom script. Each node in the graph contained features such as position, speed, and direction. The GraphSAGE algorithm was implemented to generate node embeddings, which were then passed into a Transformer encoder block consisting of multi-head attention layers, feed-forward networks, and layer normalization. The final classification layer predicted one of four behavior classes. Training was conducted with cross-entropy loss and evaluated using classification accuracy, F1score, and confusion matrices.

Create models

Model creation was approached using a structured and iterative methodology to ensure that each subcomponent met its functional and performance goals before integration. The process began by defining the architecture, selecting hyperparameters, and preparing the data pipelines required for training. All models were implemented using open-source libraries—YOLOv5 in PyTorch, the CNN model in TensorFlow/Keras, and the GNN + Transformer model in custom Tensorflow modules.

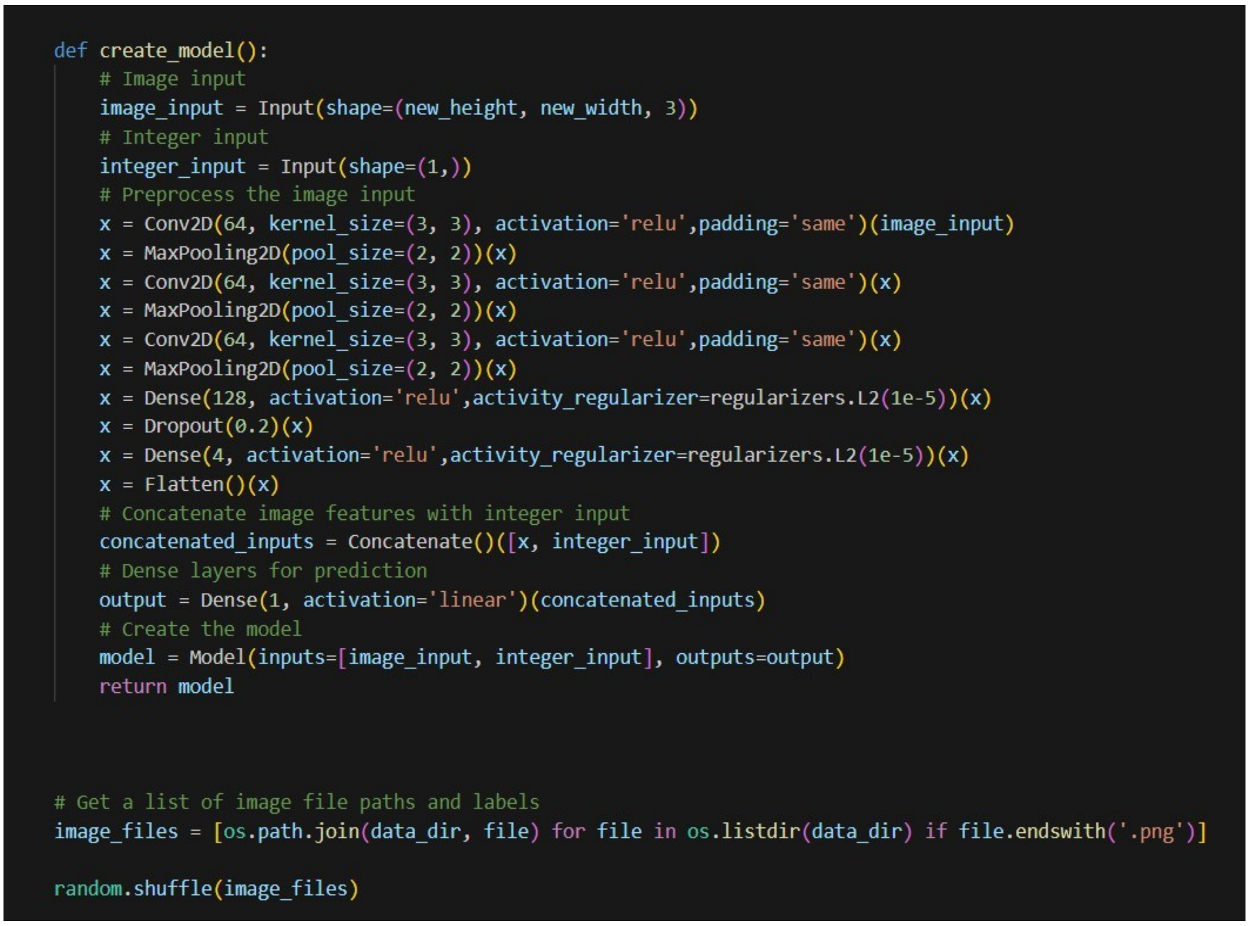
The object detection model was built using YOLOv5s, which is a compact yet powerful architecture. Initially, the pretrained weights (trained on COCO dataset) were loaded, and the model was modified to accommodate the number of custom classes in the CARLA simulation dataset. Anchors were auto-calculated based on the training data distribution, and hyperparameters such as learning rate, momentum, and weight decay were fine-tuned using the YOLOv5 training configuration file. Training was carried out for 100 epochs, and the final model was exported as a .pt file suitable for real-time inference.



```yaml
# parameters
nc: {num_classes}  # number of classes
depth_multiple: 0.33  # model depth multiple
width_multiple: 0.50  # layer channel multiple

# anchors
anchors:
  - [10,13, 16,30, 33,23]  # P3/8
  - [30,61, 62,45, 59,119]  # P4/16
  - [116,90, 156,198, 373,326]  # P5/32

# YOLOv5 backbone
backbone:
  # [from, number, module, args]
  [[-1, 1, Focus, [64, 3]],  # 0-P1/2
   [-1, 1, Conv, [128, 3, 2]],  # 1-P2/4
   [-1, 3, BottleneckCSP, [128]],
   [-1, 1, Conv, [256, 3, 2]],  # 3-P3/8
   [-1, 9, BottleneckCSP, [256]],
   [-1, 1, Conv, [512, 3, 2]],  # 5-P4/16
   [-1, 9, BottleneckCSP, [512]],
   [-1, 1, Conv, [1024, 3, 2]],  # 7-P5/32
   [-1, 1, SPP, [1024, [5, 9, 13]]],
   [-1, 3, BottleneckCSP, [1024, False]],  # 9
  ]

# YOLOv5 head
head:
  [[-1, 1, Conv, [512, 1, 1]],
   [-1, 1, nn.Upsample, [None, 2, 'nearest']],
   [[-1, 6], 1, Concat, [1]],  # cat backbone P4
   [-1, 3, BottleneckCSP, [512, False]],  # 13
```

Figure 3 yolov5s_manthrax_detection. yaml

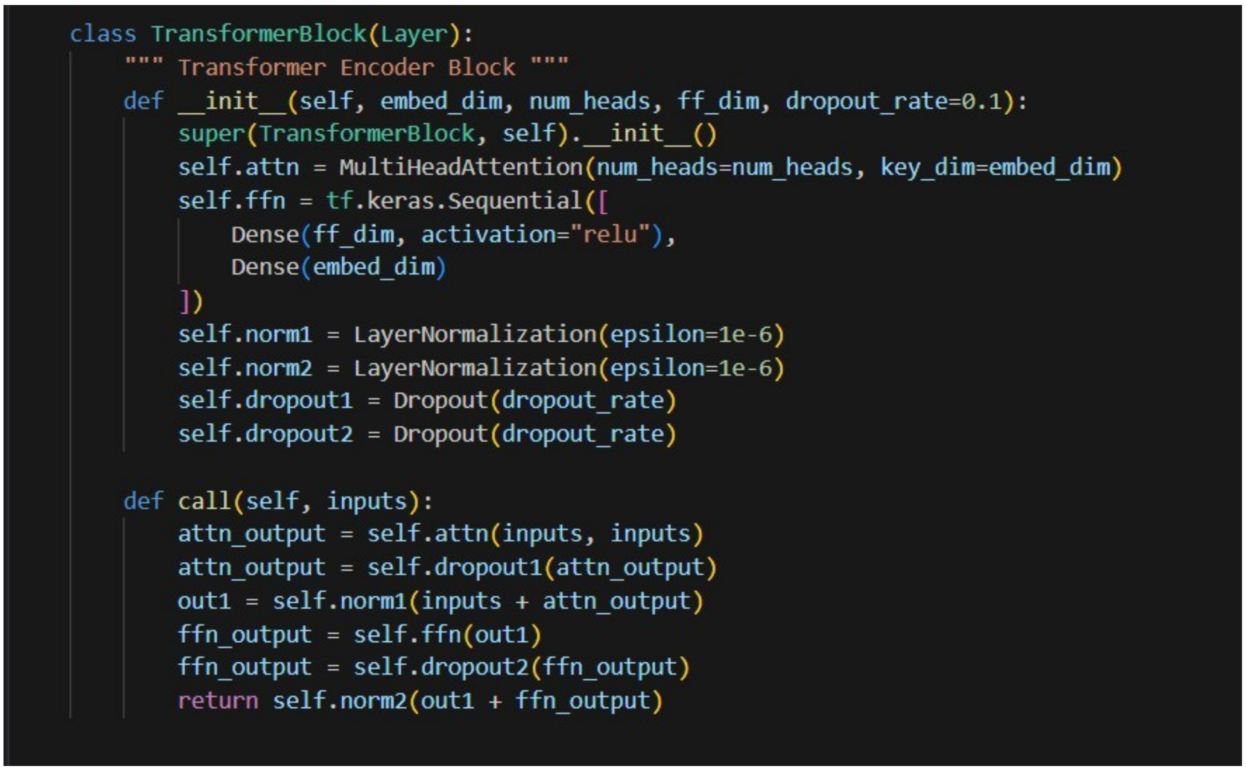
The CNN Lane keeping model was implemented using a sequential architecture comprising convolutional, activation, dropout, and fully connected layers. The image input shape was fixed at 320x180 pixels. Each training image was paired with a numerical label representing the steering angle. The loss function used was Mean Squared Error (MSE), and the training process utilized the Adam optimizer with early stopping enabled to avoid overfitting. The final model was saved in HDF5 format (.h5) and evaluated using validation loss and steering angle deviation metrics



```python
def create_model():
    # Image input
    image_input = Input(shape=(new_height, new_width, 3))
    # Integer input
    integer_input = Input(shape=(1,))
    # Preprocess the image input
    x = Conv2D(64, kernel_size=(3, 3), activation='relu',padding='same')(image_input)
    x = MaxPooling2D(pool_size=(2, 2))(x)
    x = Conv2D(64, kernel_size=(3, 3), activation='relu',padding='same')(x)
    x = MaxPooling2D(pool_size=(2, 2))(x)
    x = Conv2D(64, kernel_size=(3, 3), activation='relu',padding='same')(x)
    x = MaxPooling2D(pool_size=(2, 2))(x)
    x = Dense(128, activation='relu',activity_regularizer=regularizers.L2(1e-5))(x)
    x = Dropout(0.2)(x)
    x = Dense(4, activation='relu',activity_regularizer=regularizers.L2(1e-5))(x)
    x = Flatten()(x)
    # Concatenate image features with integer input
    concatenated_inputs = Concatenate()([x, integer_input])
    # Dense layers for prediction
    output = Dense(1, activation='linear')(concatenated_inputs)
    # Create the model
    model = Model(inputs=[image_input, integer_input], outputs=output)
    return model



# Get a list of image file paths and labels
image_files = [os.path.join(data_dir, file) for file in os.listdir(data_dir) if file.endswith('.png')]

random.shuffle(image_files)
```

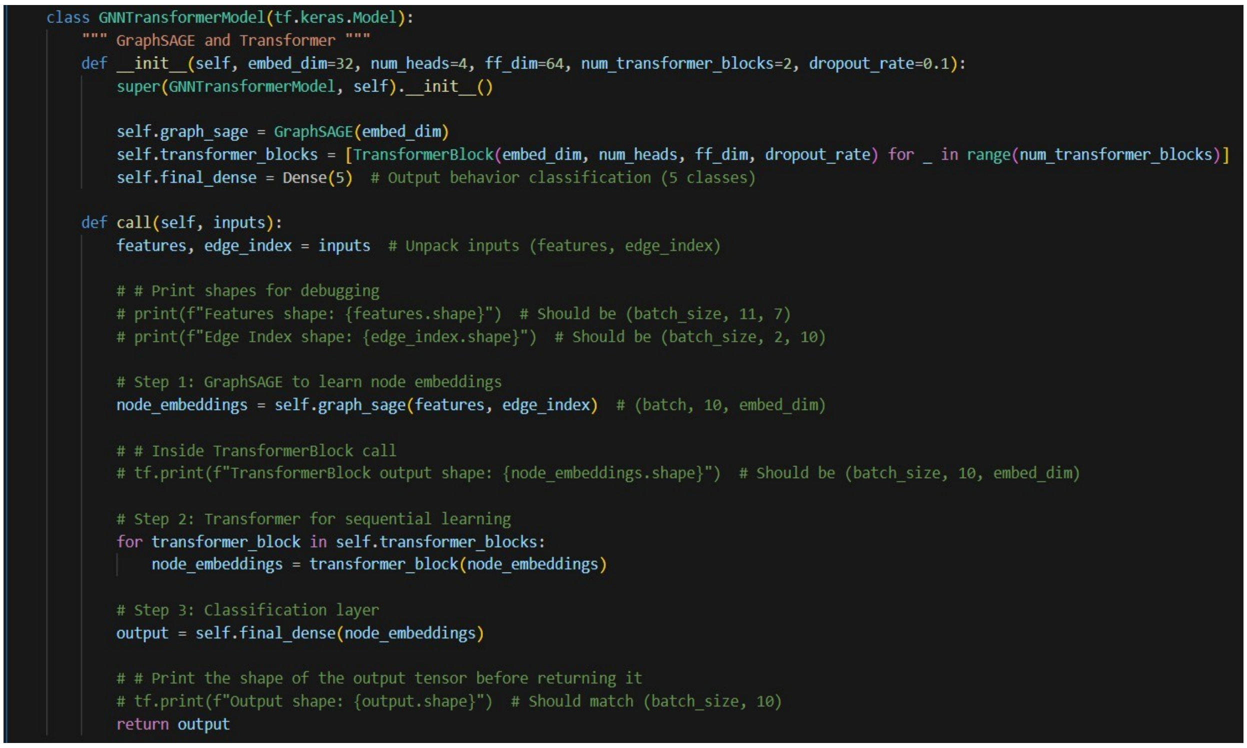
Figure 4 CNN Model Creation

This For the GNN + Transformer model, graph creation was handled by a custom preprocessing script that generated node-feature dictionaries and adjacency matrices from sequential object position logs. GraphSAGE was used as the GNN layer to compute embeddings, followed by Transformer Encoder blocks with multi-head self-attention. The combined embeddings were passed through a fully connected layer with softmax activation to classify each agent’s behavior into four categories. The model was compiled with a categorical cross-entropy loss function and accuracy as the main evaluation metric. After training, the model was saved both as a standard TensorFlow/Keras model and exported to TensorFlow Lite format for compatibility with edge devices.



```python
class TransformerBlock(Layer):
    """ Transformer Encoder Block """
    def __init__(self, embed_dim, num_heads, ff_dim, dropout_rate=0.1):
        super(TransformerBlock, self).__init__()
        self.attn = MultiHeadAttention(num_heads=num_heads, key_dim=embed_dim)
        self.ffn = tf.keras.Sequential([
            Dense(ff_dim, activation="relu"),
            Dense(embed_dim)
        ])
        self.norm1 = LayerNormalization(epsilon=1e-6)
        self.norm2 = LayerNormalization(epsilon=1e-6)
        self.dropout1 = Dropout(dropout_rate)
        self.dropout2 = Dropout(dropout_rate)

    def call(self, inputs):
        attn_output = self.attn(inputs, inputs)
        attn_output = self.dropout1(attn_output)
        out1 = self.norm1(inputs + attn_output)
        ffn_output = self.ffn(out1)
        ffn_output = self.dropout2(ffn_output)
        return self.norm2(out1 + ffn_output)
```

Figure 5 Transformer Encoder Block



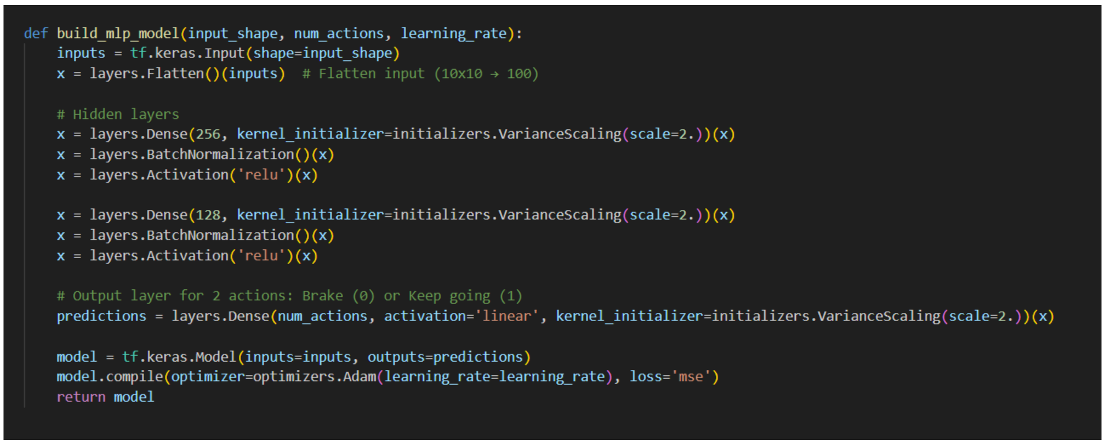
```python
class GNNTransformerModel(tf.keras.Model):
    """ GraphSAGE and Transformer """
    def __init__(self, embed_dim=32, num_heads=4, ff_dim=64, num_transformer_blocks=2, dropout_rate=0.1):
        super(GNNTransformerModel, self).__init__()

        self.graph_sage = GraphSAGE(embed_dim)
        self.transformer_blocks = [TransformerBlock(embed_dim, num_heads, ff_dim, dropout_rate) for _ in range(num_transformer_blocks)]
        self.final_dense = Dense(5)  # Output behavior classification (5 classes)

    def call(self, inputs):
        features, edge_index = inputs  # Unpack inputs (features, edge_index)

        # # Print shapes for debugging
        # print(f"Features shape: {features.shape}")  # Should be (batch_size, 11, 7)
        # print(f"Edge Index shape: {edge_index.shape}")  # Should be (batch_size, 2, 10)

        # Step 1: GraphSAGE to learn node embeddings
        node_embeddings = self.graph_sage(features, edge_index)  # (batch, 10, embed_dim)

        # # Inside TransformerBlock call
        # tf.print(f"TransformerBlock output shape: {node_embeddings.shape}")  # Should be (batch_size, 10, embed_dim)

        # Step 2: Transformer for sequential learning
        for transformer_block in self.transformer_blocks:
            node_embeddings = transformer_block(node_embeddings)

        # Step 3: Classification layer
        output = self.final_dense(node_embeddings)

        # # Print the shape of the output tensor before returning it
        # tf.print(f"Output shape: {output.shape}")  # Should match (batch_size, 10)
        return output
```

Figure 6 GNN + Transformer hybrid model

2.6.1.2 Decision Making and Collison Avoidance

The implementation phase of the MANTHRA-X system involved constructing and integrating a hybrid Deep Reinforcement Learning (DRL)-based decision-making model in combination with traditional PID control into the CARLA simulator. The objective was to create a reliable, efficient collision avoidance system that is capable of making intelligent driving decisions in real time.

The CARLA simulator served as the development and testing platform, offering diverse driving scenarios such as urban intersections, highways, pedestrian zones, and obstacle-dense routes. The simulation platform had several sensor feeds LiDAR, RGB cameras, GPS, and semantic segmentation that were used to construct the vehicle's perception of the world.



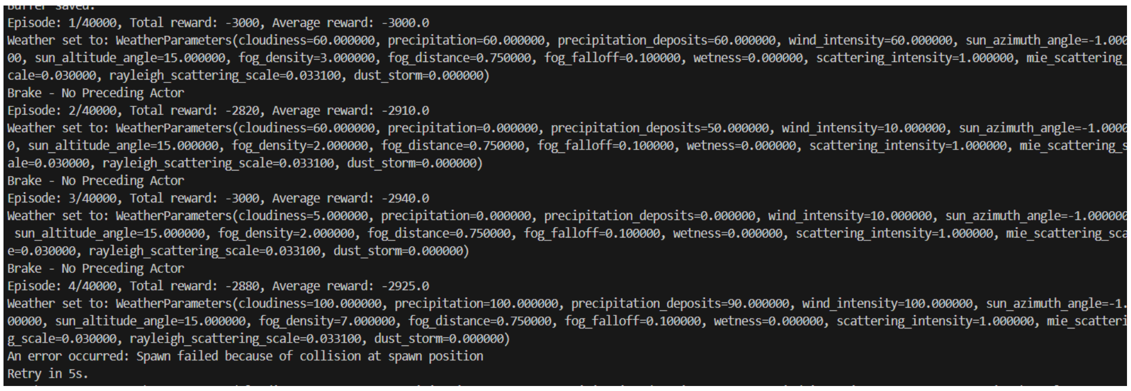
```python
def build_mlp_model(input_shape, num_actions, learning_rate):
    inputs = tf.keras.Input(shape=input_shape)
    x = layers.Flatten()(inputs)  # Flatten input (10x10 → 100)

    # Hidden layers
    x = layers.Dense(256, kernel_initializer=initializers.VarianceScaling(scale=2.))(x)
    x = layers.BatchNormalization()(x)
    x = layers.Activation('relu')(x)

    x = layers.Dense(128, kernel_initializer=initializers.VarianceScaling(scale=2.))(x)
    x = layers.BatchNormalization()(x)
    x = layers.Activation('relu')(x)

    # Output layer for 2 actions: Brake (0) or Keep going (1)
    predictions = layers.Dense(num_actions, activation='linear', kernel_initializer=initializers.VarianceScaling(scale=2.))(x)

    model = tf.keras.Model(inputs=inputs, outputs=predictions)
    model.compile(optimizer=optimizers.Adam(learning_rate=learning_rate), loss='mse')
    return model
```

Figure 7 DQN model implementation

The DQN model was implemented in PyTorch and trained to learn high-level control policies such as lane changes, acceleration, deceleration, and braking. It was trained using a custom reward function that encouraged safe driving practices and discouraged unsafe or abrupt behavior. Training was performed using curriculum learning and randomized scenario generation to promote generalization.

DQN Functionality:

- **Inputs (state space)**: Position, speed, obstacle distances, lane deviation, traffic light state, relative velocities.
- **Actions**: High-level decisions such as "turn left", "slow down", "change lane", or "brake".
- **Rewards**: Negative for collisions, harsh braking, or crossing lane boundaries; positive for smooth navigation, speed maintenance, and safe distance keeping.



```
Buffer saved.
Episode: 1/40000, Total reward: -3000, Average reward: -3000.0
Weather set to: WeatherParameters(cloudiness=60.000000, precipitation=60.000000, precipitation_deposits=60.000000, wind_intensity=60.000000, sun_azimuth_angle=-1.000
00, sun_altitude_angle=15.000000, fog_density=3.000000, fog_distance=0.750000, fog_falloff=0.100000, wetness=0.000000, scattering_intensity=1.000000, mie_scattering_
cale=0.030000, rayleigh_scattering_scale=0.033100, dust_storm=0.000000)
Brake - No Preceding Actor
Episode: 2/40000, Total reward: -2820, Average reward: -2910.0
Weather set to: WeatherParameters(cloudiness=60.000000, precipitation=0.000000, precipitation_deposits=50.000000, wind_intensity=10.000000, sun_azimuth_angle=-1.0000
0, sun_altitude_angle=15.000000, fog_density=2.000000, fog_distance=0.750000, fog_falloff=0.100000, wetness=0.000000, scattering_intensity=1.000000, mie_scattering_s
ale=0.030000, rayleigh_scattering_scale=0.033100, dust_storm=0.000000)
Brake - No Preceding Actor
Episode: 3/40000, Total reward: -3000, Average reward: -2940.0
Weather set to: WeatherParameters(cloudiness=5.000000, precipitation=0.000000, precipitation_deposits=0.000000, wind_intensity=10.000000, sun_azimuth_angle=-1.000000
 sun_altitude_angle=15.000000, fog_density=2.000000, fog_distance=0.750000, fog_falloff=0.100000, wetness=0.000000, scattering_intensity=1.000000, mie_scattering_sca
e=0.030000, rayleigh_scattering_scale=0.033100, dust_storm=0.000000)
Brake - No Preceding Actor
Episode: 4/40000, Total reward: -2880, Average reward: -2925.0
Weather set to: WeatherParameters(cloudiness=100.000000, precipitation=100.000000, precipitation_deposits=90.000000, wind_intensity=100.000000, sun_azimuth_angle=-1.
00000, sun_altitude_angle=15.000000, fog_density=7.000000, fog_distance=0.750000, fog_falloff=0.100000, wetness=0.000000, scattering_intensity=1.000000, mie_scatteri
g_scale=0.030000, rayleigh_scattering_scale=0.033100, dust_storm=0.000000)
An error occurred: Spawn failed because of collision at spawn position
Retry in 5s.
```

Figure 8 Reward output of training process

Reward functions:

Table 2 Reward and Penalty List

Reward Function	Condition	Reward Value	Description
Terminal Reward	Collision occurs	-5000	Large penalty for collisions.
	No collisions and episode ends successfully	+1000	Large reward for successful episode completion.
Speeding Reward	No preceding vehicle or safe distance maintained	+60	Reward for maintaining speed without obstacles.

	Preceding vehicle is too close (distance < 12)	-300	Penalty for getting too close to the preceding vehicle.
	Preceding vehicle is at a safe distance (distance ≥ 12)	+30	Reward for maintaining a safe distance.
Brake Reward	No preceding vehicle	-3000	Penalty for braking when no preceding vehicle is present.
	Preceding vehicle is too far (distance > 12)	-1500	Penalty for braking too far from the preceding vehicle.
	Preceding vehicle is too close (distance < 6)	-700	Penalty for braking too close to the preceding vehicle.
	Preceding vehicle is at an optimal distance (6 ≤ distance ≤ 12)	+2000	Reward for braking at an optimal distance.
Episode End	No collisions and episode ends successfully	+1000	Large reward for successful episode completion.

- Training Protocol:
    - Curriculum learning: Progressively increase scenario difficulty
    - Transfer learning: Pre-train in simulation, can be fine tuned with real world data

For low-level actuation, a PID controller was included to allow smooth execution of DQN decisions. Steering, throttle, and braking were controlled by the PID to find a balance between responsiveness and stability. The hybrid DQN-PID architecture allowed the system to benefit from learning-based adaptability and deterministic control precision.

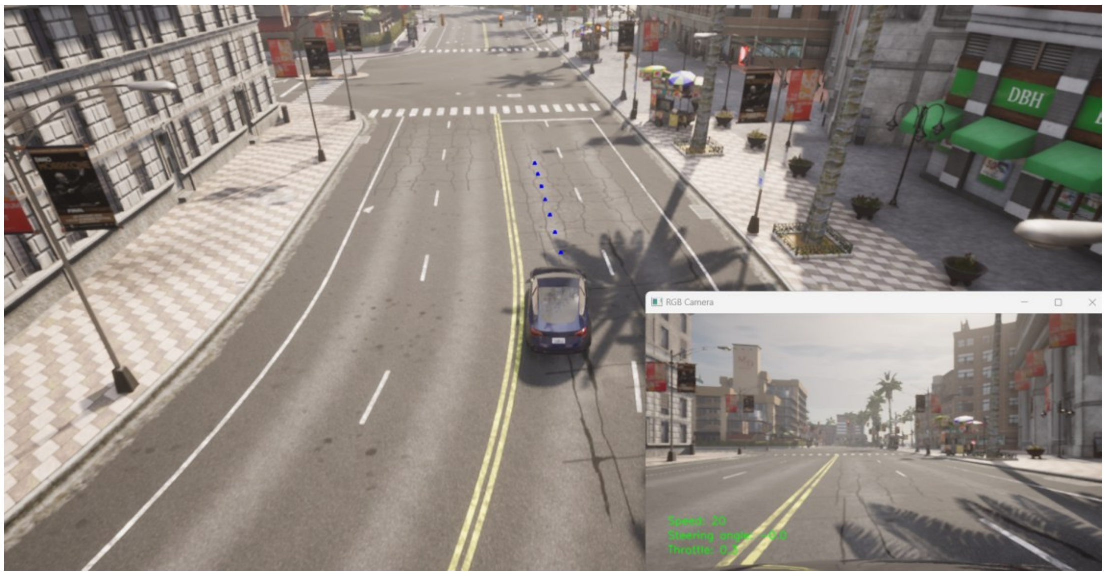


Figure 9 PID output in CARLA

To enhance safety, a collision avoidance module was incorporated, which continuously tracked the future trajectory of the vehicle and its surroundings. When there was a possibility of collision, this module overrode DQN's decision to take emergency measures such as braking or lane change.

They also used a mobile alerting system to provide real-time feedback. Developed using Flutter, the app displayed telemetry data from the AV system and gave immediate alerts if the model detected high-risk scenarios. Alerts included information on obstacle type, time to collision, and recommended actions, giving an extra layer of transparency and user engagement.

The system was tested for real-time performance with stable behavior above 30 FPS on an RTX 4060 GPU. Additional optimization with model pruning and edge deployment techniques ensured that the architecture can perform even on constrained hardware like NVIDIA Jetson devices.

2.6.1.3 Ethical decision making and driver awareness monitoring in autonomous vehicles

Dataset Collection and Preprocessing

The Driver Awareness Monitoring system used a mix of public datasets and custom video recordings. Public datasets included annotated driver behaviors like focus, drowsiness, and distractions for initial algorithm tuning. To supplement this, custom data was collected using webcams and Raspberry Pi devices in varied environments, capturing simulated behaviors such as yawning and head movements for broader generalization.

Video frames were processed using Python and OpenCV, extracting 468 facial landmarks via MediaPipe Face Mesh, with emphasis on eye and iris regions. From these, features like Eye Aspect Ratio (EAR), blink rates, and iris movement were calculated. Head pose data further refined gaze classification. The preprocessing also included brightness normalization, noise filtering, and temporal smoothing to enhance performance under real-world conditions.

For the ethical module, synthetic traffic scenarios involving moral dilemmas were created using OpenAI Gym-style environments. These simulated intersections and obstacle-rich situations provided diverse inputs for the DQN model. The agent gathered interaction data as (state, action, reward, next_state, done) tuples, which were stored in replay buffers and used for iterative model training and validation.

Ethical Decision-Making Model

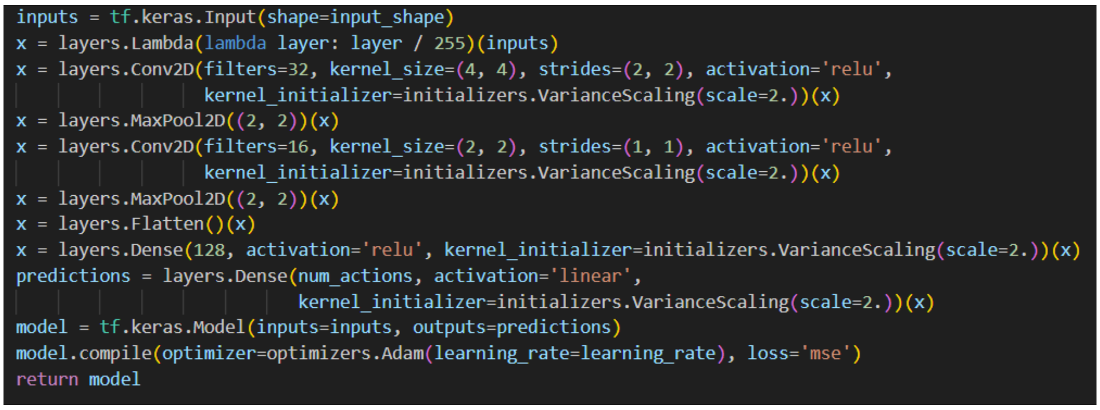
Reinforcement Learning Approach (DQN**)**

The ethical decision-making framework proposed in this research is centered around a Deep Q-Network (DQN) algorithm implemented using TensorFlow. This architecture is specially tailored for autonomous vehicle (AV) control tasks involving ethical dilemmas in complex environments. DQN combines reinforcement learning principles with deep convolutional neural networks (CNNs) to approximate Q-values for state-action pairs, enabling the AV to learn optimal decision-making policies through simulated experiences.

The system architecture includes a Policy Network (CNN estimating Q-values for decision-making), a Target Network (a synchronized duplicate for stable training), a Replay Buffer (storing experience tuples for mini-batch sampling), and an ε-greedy policy (balancing exploration and exploitation through epsilon decay).

The CNN architecture for the policy and target networks consists of:

- Input layer: Accepts 2D input (e.g., simulation frames) and normalizes pixel values to [0,1].
- Convolutional layers: Two Conv2D layers with ReLU activation and VarianceScaling kernel initialization to improve convergence.
- Pooling layers: MaxPool layers following each convolutional layer to reduce dimensionality and extract key spatial features.
- Dense layers: A 128-unit fully connected layer with ReLU activation.
- Output layer: A dense layer with linear activation returning Q-values for each discrete action.



```
inputs = tf.keras.Input(shape=input_shape)
x = layers.Lambda(lambda layer: layer / 255)(inputs)
x = layers.Conv2D(filters=32, kernel_size=(4, 4), strides=(2, 2), activation='relu',
                  kernel_initializer=initializers.VarianceScaling(scale=2.))(x)
x = layers.MaxPool2D((2, 2))(x)
x = layers.Conv2D(filters=16, kernel_size=(2, 2), strides=(1, 1), activation='relu',
                  kernel_initializer=initializers.VarianceScaling(scale=2.))(x)
x = layers.MaxPool2D((2, 2))(x)
x = layers.Flatten()(x)
x = layers.Dense(128, activation='relu', kernel_initializer=initializers.VarianceScaling(scale=2.))(x)
predictions = layers.Dense(num_actions, activation='linear',
                           kernel_initializer=initializers.VarianceScaling(scale=2.))(x)
model = tf.keras.Model(inputs=inputs, outputs=predictions)
model.compile(optimizer=optimizers.Adam(learning_rate=learning_rate), loss='mse')
return model
```

Figure 10 CNN Architecture

Additional engineering considerations include:

- The use of tf.keras.optimizers.Adam with a specified learning rate.

- Model saving and loading functionality for both policy weights and replay buffer state.
- Evaluation-only policy method (act_trained) that excludes exploration for reproducible ethical behavior during testing.
- TensorBoard logging via tf.keras.callbacks.TensorBoard and tf.summary for visual monitoring of training progress and debugging.
- Dynamic ethical parameterization using knob_value to tune the agent's ethical stance.
- Episode counters and evaluation checkpoints to support training milestones, performance monitoring, and behavior analysis.

This architecture provides a robust foundation for learning moral behavior in AVs, allowing the model to respond adaptively and transparently to ethical challenges in diverse scenarios.

Ethical Reward Function Design

A core component of this research is the design of an ethical reward function that guides the AV agent to learn behavior aligned with human-centered ethical principles. Unlike conventional AV rewards that focus on efficiency or speed, the ethical reward function incorporates metrics such as harm avoidance, passenger safety, and compliance with road rules.

A notable innovation in this design is the inclusion of the k-value parameter, which allows dynamic ethical prioritization. This parameter adjusts the weighting between egoistic (passenger-focused) and altruistic (pedestrian-focused) outcomes.

For example:

- $k = 0.1$: High priority to external safety (altruism)
- $k = 0.9$: High priority to internal safety (egoism)

This feature enables the system to simulate varying moral configurations based on cultural or regulatory demands, and is reflected directly in the reward function calculations.

Simulation Setup and Training

The ethical decision-making system is trained in a custom simulation environment that replicates complex urban driving scenarios with ethical challenges, such as unpredictable pedestrians, traffic signals, intersections, and dynamic hazards. Each episode begins with the AV in a randomized state, and the agent receives spatial-temporal state inputs to select actions (e.g., accelerate, brake, steer) guided by a human-centered ethical reward function. These interactions generate (state, action, reward, next_state, done) tuples stored in a replay buffer. The DQN model is trained over thousands of episodes using mini-batch sampling and mean-squared error loss, with periodic target network updates to ensure stability. An ε-greedy policy balances exploration and exploitation. Training metrics—including cumulative reward, collision rate, ethical score, and loss—are logged using TensorBoard, while checkpointing allows for regular saving of model weights and replay data for retraining and analysis.

Driver Awareness Monitoring

To enhance human-centered safety in semi-autonomous driving, the system incorporates a real-time Driver Awareness Monitoring module, designed for deployment on embedded platforms such as Raspberry Pi. This module combines lightweight computer vision techniques, multithreaded video processing, facial behavior analysis, and IoT-based alert mechanisms to detect and respond to driver drowsiness, distraction, and absence.

<u>Eyeball Tracking Using MediaPipe</u>

The system employs Google MediaPipe's Face Mesh, extracting 468 facial landmarks in real time, including those around the eyes and irises. OpenCV is used for video frame capture on Raspberry Pi. Captured frames are preprocessed (flipped and converted to RGB) before inference.

<u>Gaze Zone Classification</u>

Gaze direction is classified into central (focused), peripheral, and off-road zones by measuring horizontal iris displacement relative to the eye corners. Additional 3D cues, such as head tilt and z-coordinate shifts, are incorporated to improve focus detection.

<u>EAR-Based Drowsiness Detection</u>

The Eye Aspect Ratio (EAR) is calculated from vertical and horizontal eye distances. A threshold-based logic flags drowsiness when the EAR remains below a predefined value (e.g., 0.2) for a consecutive number of frames. Combined with gaze tracking and face landmark visibility, this allows the system to classify:

- EAR $< 0.2 \rightarrow$ Drowsy
- Iris deviation $\rightarrow$ Distracted
- No landmarks $\rightarrow$ Not detected

A rolling window mechanism ensures consistent detection before triggering alerts.

Multithreaded Inference System

The monitoring pipeline operates using Python's threading module. Two concurrent threads manage:

- `capture_frames()`: Continuous frame acquisition.
- `drowsiness_detection_system()`: Real-time frame analysis.

Shared frame data is protected using a threading lock to ensure synchronization and processing efficiency.

IoT Integration and Alert Escalation

The system issues alerts based on sustained detection over a threshold number of frames. Defined alert codes include:

- `0`: Driver Not Detected
- `1`: Drowsy
- `2`: Distracted

Alerts are sent to a Firebase backend, enabling real-time communication with a paired mobile application.

GPIO-Controlled Buzzer for Local Alerts

A buzzer connected to the Raspberry Pi's GPIO acts as a redundant local alert mechanism. On alert trigger, the buzzer provides haptic feedback, ensuring the driver is notified even without cloud or mobile connectivity.

Mobile Application Integration

To complement the driver monitoring system, a dedicated mobile application has been developed to deliver real-time alerts, enhance situational awareness, and support postevent analysis. The application is implemented using Flutter, Android SDK (Java/Kotlin), and Firebase Realtime Database, offering a responsive and scalable interface suitable for both individual drivers and fleet management operations.

Architecture and Connectivity

The mobile application establishes a persistent connection to the Firebase Realtime Database, subscribing to the "IOT/latest_alert" node, where alerts generated by the Raspberry Pi module are uploaded in real time. Each alert is identified by a unique warning_id corresponding to the detected driver state:

- 5: Driver Not Detected
- 6: Drowsiness Detected
- 7: Driver is Not Focused

To prevent redundant processing, an internal cache (_lastWarningId) ensures that duplicate alerts are ignored. On detecting a new event, the app logs it to the "iot_alerts" node for historical record-keeping and simultaneously triggers a notification using the FlutterLocalNotificationsPlugin.

The application follows a modular architecture comprising:

- Real-time alert listener
- Local notification engine
- Alert history manager

This separation allows seamless extension to multi-vehicle dashboards and integration with enterprise fleet monitoring platforms.

Real-Time Alerts: Visual and Haptic Feedback

Upon detecting a critical driver state, the application triggers immediate visual and haptic alerts using high-priority native Android notification channels. These notifications are configured to maximize driver awareness under various environmental conditions.

Key features include:

- Customized push notifications (e.g., *“Frequent Drowsiness Detected!”*)

- High-priority alert channels with vibration and sound
- Custom vibration patterns for high-severity events
- On-screen visual warnings and full-screen overlays (optional)

SMS Escalation Logic

For heightened safety in fleet environments, the system supports SMS escalation when repeated alerts occur. If five alerts of the same type are detected within a short time window, an SMS is sent to designated contacts such as fleet managers or vehicle owners. This logic can be implemented using Firebase Functions or integrated with third-party services such as Twilio, enhancing accountability and rapid incident response.

2.6.1.4 An Intelligent Multi-Modal In-Cabin Threat Detection System for Level 3 Autonomous Vehicles

Data Collection & Annotation: Finalize datasets for training and validation, ensuring balance across threat types, voice tones, emotional expressions, and lighting conditions.

Model Training: Train models separately (YOLOv8, Audio CNN, Emotion model) using preprocessed and augmented datasets, followed by cross-validation.

Modular Integration: Merge all three models into a unified decision pipeline with timestamp synchronization and rule-based logic. 40

Edge Optimization: Convert models to ONNX/TensorRT for deployment on embedded systems with low power usage and high efficiency.

Front-End Development: Implement user interface for alert configuration, logs review, and threat visualization.

System Integration: Combine hardware modules (camera, mic, edge board) into a compact, mountable system for cabin installation.

Field Testing: Deploy in real-world vehicles for limited routes, collecting logs, user feedback, and system error rates.

2.6.1.5 Tools, Technologies, and Algorithms

The development of the autonomous vehicle support framework integrated a diverse range of tools and technologies across machine learning, computer vision, embedded systems, mobile development, and cloud integration.

Programming Languages:

- **Python** powered ML models, hardware interfacing, and real-time video analysis.
- **Dart (Flutter)** enabled a cross-platform mobile app with real-time UI updates.
- **Java/Kotlin** supported native Android features like notifications and vibration.

AI/ML Frameworks:

- **TensorFlow & Keras** were used for DQN-based ethical decision-making and CNN models.
- **PyTorch & YOLOv5/YOLOv8** enabled fast object detection and audio classification.
- **TensorFlow Geometric** and **Transformer** models handled graph and sequence learning.

Computer Vision:

- **OpenCV** and **MediaPipe** enabled facial landmark detection, EAR calculation, and real-time video processing for driver awareness.

Hardware Platforms:

- **Raspberry Pi** served as the edge device for real-time inference and alerting.
- **Webcam** captured facial data, and **GPIO** controlled buzzer alerts.

Mobile & Cloud Integration:

- **Flutter** developed the app, with **Firebase Realtime Database** enabling alert sync.
- **Local Notifications** and optional **SMS** escalation ensured effective alerting.

Backend and Streaming:

- **Flask REST API** and **Apache Kafka** (for future scaling) supported backend logic and message brokering.

Key Algorithms:

- **DQN** with replay buffer, target network, and epsilon-greedy policy.
- **GNN + Transformer**, CNN, and NMS for perception and decision modeling.

2.6.2 Testing

2.6.2.1 Object detection and motion prediction in autonomous vehicles

The testing phase was an essential component of the development cycle for validating the performance, reliability, and robustness of the Manthra-X perception module. Testing was conducted both at the component level and at the system integration level within the CARLA simulator. The objective was to assess how well each individual model performed in isolation and how effectively the integrated system handled diverse driving scenarios in real-time. The YOLOv5 object detection model was tested on a dedicated subset of simulated frames encompassing diverse lighting, weather, and occlusion scenarios. Key performance indicators— such as precision, recall, mean Average Precision at 0.5 Intersection over Union (mAP@0.5), and inference speed measured in frames per second (FPS)—were computed. The model achieved high accuracy for commonly detected classes like vehicles and pedestrians, maintaining strong localization capabilities even under adverse conditions. For the CNN-based lane keeping system, evaluation was conducted using a separate set of driving frames with annotated ground truth steering angles. The model's predicted outputs were compared against actual steering values using Root Mean Squared Error (RMSE) and the coefficient of determination (R^2). The evaluation demonstrated that the CNN effectively maintained lane alignment across various scenarios, including straight paths, curves, and temporary occlusions. Additionally, visual debugging overlays were used to compare predicted and actual lane positions, with performance anomalies analyzed to guide further model refinement The GNN + Transformer motion prediction model was tested using time-sequenced data logs from multiple dynamic agents in traffic-rich scenarios. The model's classification outputs (e.g., Going, Parked, Crossing) were validated using confusion matrices, classification accuracy, and F1-scores. Behavioral transitions and misclassifications were reviewed using temporal playback within the simulation. The model consistently achieved high classification accuracy, particularly in predicting behaviors that depend on temporal and relational context. To evaluate the full perception pipeline, the models were executed in parallel within the CARLA simulation loop. This allowed the assessment of latency, synchronization, and interaction fidelity between models. The integrated system was subjected to stress-testing scenarios, such as multiple vehicle crossings, tight corners, and sudden pedestrian appearances. 30 System response time, frame drop rate, and combined output integrity were monitored and logged. step In addition to quantitative testing, qualitative validation was also performed by visualizing all outputs—bounding boxes, lane overlays, and behavior tags—on the CARLA interface and mobile app. This enabled an intuitive evaluation of system consistency, especially during edge cases. Overall, the testing process confirmed that the Manthra-X perception system met its performance expectations across all subcomponents and functioned reliably under integrated conditions. Insights from this phase also provided actionable feedback for refining model thresholds, retraining with edge-case data, and preparing for real-world validation.

2.6.2.2 Autonomous vehicle collision avoidance

a. Simulation-Based Testing of DQN Models

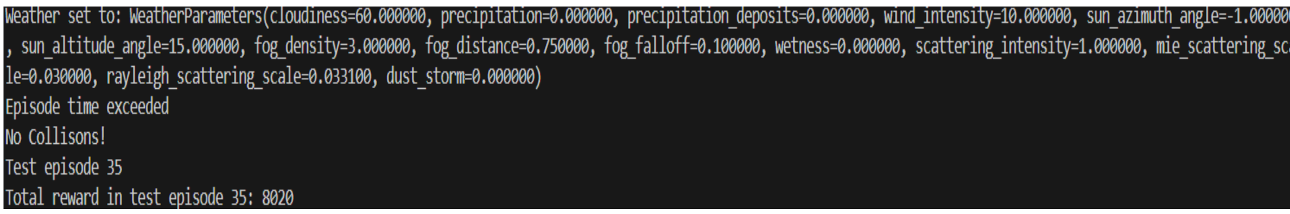
The primary testing phase involved evaluating the collision avoidance model within the CARLA simulator, which offered a safe, controlled environment for repeatable experiments. Scenarios were designed to assess the model’s ability to make safe and intelligent decisions in response to dynamic obstacles and uncertain behaviors from other road users.

The following simulation scenarios were used as test scenarios:

- Urban intersections with unpredictable pedestrian movements
- Multi-lane highways with sudden lane changes by nearby vehicles
- Congested city streets with static and dynamic obstacles
- Overtaking and merging maneuvers
- Adverse weather conditions, such as fog and rain

Each scenario was repeated multiple times with slight variations to introduce randomness and test model generalization.

b. Performance Metrics



```
Weather set to: WeatherParameters(cloudiness=60.000000, precipitation=0.000000, precipitation_deposits=0.000000, wind_intensity=10.000000, sun_azimuth_angle=-1.00000
, sun_altitude_angle=15.000000, fog_density=3.000000, fog_distance=0.750000, fog_falloff=0.100000, wetness=0.000000, scattering_intensity=1.000000, mie_scattering_sc
le=0.030000, rayleigh_scattering_scale=0.033100, dust_storm=0.000000)
Episode time exceeded
No Collisons!
Test episode 35
Total reward in test episode 35: 8020
```

Figure 11: Testing Reward Output

Model performance was assessed using the following metrics:

- Collision Rate: Percentage of episodes that resulted in a collision
- Time to Collision (TTC): Time gap before a potential collision was avoided
- Success Rate: Completion of navigation tasks without intervention
- Comfort Score: Frequency of abrupt maneuvers or oscillatory behavior
- Policy Convergence: Stability and reward consistency over training episodes

c. Model Robustness and Generalization

To assess generalization, the trained DRL models were evaluated on unseen scenarios and different traffic configurations:

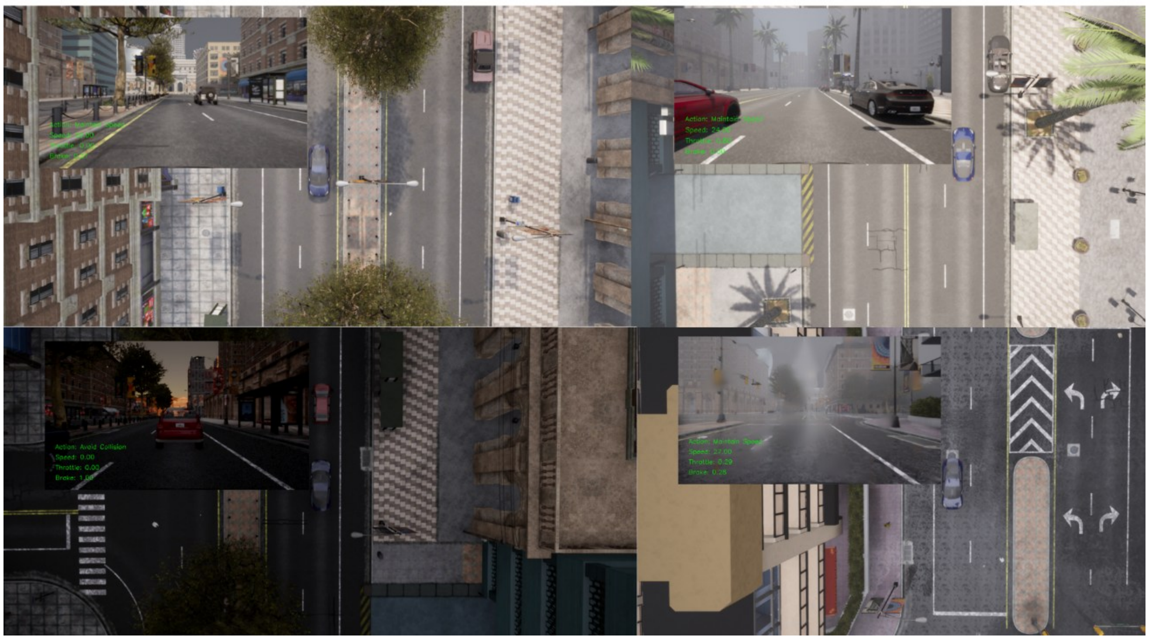


Figure 12 Random scenarios in CARLA

- Modified map layouts and road geometries

- Different vehicle types (bikes, trucks)
- Previously unseen pedestrian densities

The models retained strong performance, indicating successful generalization. However, performance slightly degraded in environments with noisy or incomplete sensor inputs, suggesting further work could focus on robust policy learning or sensor fusion techniques.

a. Testing and Validation of Mobile Application

The mobile app was tested using simulation data from CARLA, an open-source AV simulator that generates realistic driving environments and agent behaviors. Scenarios included:

- Highway driving with dynamic traffic
- Pedestrian crosswalks
- Intersection navigation
- Obstacle avoidance under varying weather conditions

The app was also stress-tested under network instability conditions to ensure graceful handling of latency and packet loss. Additionally, unit tests were written for the API calls, WebSocket communication, and UI components using Flutter's testing framework.

d. System Integration Testing

End-to-end tests were performed by integrating the AV simulation with the backend and mobile interface. These tests simulated full driving episodes where:

1. The vehicle navigated a complex route.
2. DRL agents made real-time decisions.
3. Collision risks were evaluated and visualized.
4. Events were streamed live to the mobile app.

Key observations:

- Integration worked smoothly, with synchronized updates across all components.
- The backend effectively handled large volumes of telemetry without bottlenecks.
- Mobile alerts were received with minimal lag during event triggers.

e. User and Feedback Testing

A small pilot user test was conducted with peers acting as users monitoring AV behavior:

- Users found the app intuitive and informative, especially during real-time monitoring.
- Feedback highlighted interest in additional features like manual override control, route planning, and voice alerts, which could be implemented in future versions.

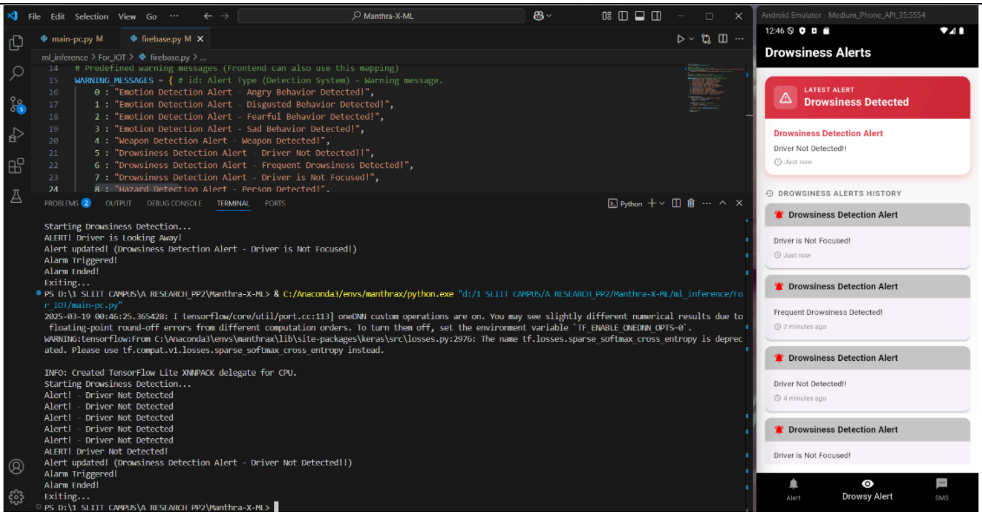
2.6.2.3 Ethical decision making and driver awareness monitoring in autonomous vehicles

Evaluation Metrics

- Accuracy of driver state detection: 91.5%
- False alert rate: < 5%
- Decision latency (Ethical engine): ~100ms per decision cycle
- Driver alert delivery latency: < 200ms
- Mobile notification delay: ~70ms

Test Cases and Scenario

Table 3 Test Case No 1

Test Case No 1	
Description	Driver not detected in camera
Expected output	Alert Code sent to Firebase and buzzer triggered, Alert shown, phone vibrated
Actual Outcome	
	
Status	Pass

```python
# Predefined warning messages (Frontend can also use this mapping)
WARNING_MESSAGES = { # id: Alert Type (Detection System) - Warning message.
    0 : "Emotion Detection Alert - Angry Behavior Detected!",
    1 : "Emotion Detection Alert - Disgusted Behavior Detected!",
    2 : "Emotion Detection Alert - Fearful Behavior Detected!",
    3 : "Emotion Detection Alert - Sad Behavior Detected!",
    4 : "Weapon Detection Alert - Weapon Detected!",
    5 : "Drowsiness Detection Alert - Driver Not Detected!!",
    6 : "Drowsiness Detection Alert - Frequent Drowsiness Detected!",
    7 : "Drowsiness Detection Alert - Driver is Not Focused!",
    8 : "Hazard Detection Alert - Person Detected!",
```

```
Starting Drowsiness Detection...
ALERT! Driver is Looking Away!
Alert updated! (Drowsiness Detection Alert - Driver is Not Focused!)
Alarm Triggered!
Alarm Ended!
Exiting...
PS D:\1 SLIIT CAMPUS\A RESEARCH_PP2\Manthra-X-ML> & C:/Anaconda3/envs/manthrax/python.exe "d:/1 SLIIT CAMPUS/A RESEARCH_PP2/Manthra-X-ML/ml_inference/Fo
r_IOT/main-pc.py"
2025-03-19 00:46:25.365428: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly different numerical results due to
 floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
WARNING:tensorflow:From C:\Anaconda3\envs\manthrax\lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprec
ated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.

INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
Starting Drowsiness Detection...
Alert! - Driver Not Detected
Alert! - Driver Not Detected
Alert! - Driver Not Detected
Alert! - Driver Not Detected
Alert! - Driver Not Detected
ALERT! Driver Not Detected!
Alert updated! (Drowsiness Detection Alert - Driver Not Detected!!)
Alarm Triggered!
Alarm Ended!
Exiting...
PS D:\1 SLIIT CAMPUS\A RESEARCH_PP2\Manthra-X-ML>
```

Drowsiness Alerts

LATEST ALERT
Drowsiness Detected
Drowsiness Detection Alert
Driver Not Detected!!
Just now

DROWSINESS ALERTS HISTORY

Drowsiness Detection Alert
Driver is Not Focused!
Just now

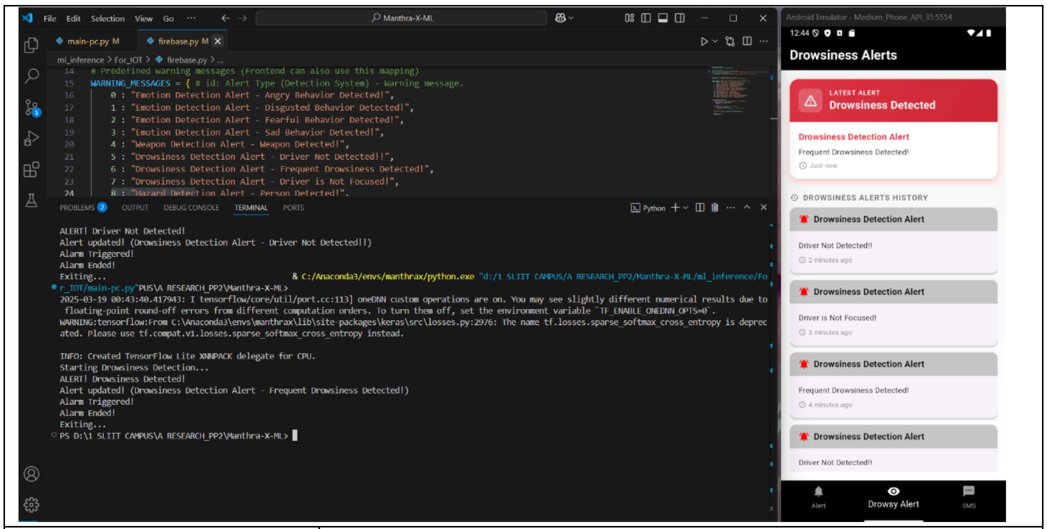
Drowsiness Detection Alert
Frequent Drowsiness Detected!
2 minutes ago

Drowsiness Detection Alert
Driver Not Detected!!
4 minutes ago

Drowsiness Detection Alert
Driver is Not Focused!

Table 4 Test Case No 2

Test Case No 2	
Description	Eyes closed for >2 seconds (drowsiness)
Expected output	Alert Code and haptic feedback on mobile
Actual Outcome	

	
Status	Pass

```python
    # Predefined warning messages (Frontend can also use this mapping)
    WARNING_MESSAGES = { # id: Alert Type (Detection System) - Warning message.
        0 : "Emotion Detection Alert - Angry Behavior Detected!",
        1 : "Emotion Detection Alert - Disgusted Behavior Detected!",
        2 : "Emotion Detection Alert - Fearful Behavior Detected!",
        3 : "Emotion Detection Alert - Sad Behavior Detected!",
        4 : "Weapon Detection Alert - Weapon Detected!",
        5 : "Drowsiness Detection Alert - Driver Not Detected!!",
        6 : "Drowsiness Detection Alert - Frequent Drowsiness Detected!",
        7 : "Drowsiness Detection Alert - Driver is Not Focused!",
        8 : "Hazard Detection Alert - Person Detected!",
```

```
ALERT! Driver Not Detected!
Alert updated! (Drowsiness Detection Alert - Driver Not Detected!!)
Alarm Triggered!
Alarm Ended!
Exiting...                          & C:/Anaconda3/envs/manthrax/python.exe "d:/1 SLIIT CAMPUS/A RESEARCH_PP2/Manthra-X-ML/ml_inference/Fo
r_IOT/main-pc.py"PUS\A RESEARCH_PP2\Manthra-X-ML>
2025-03-19 00:43:40.417943: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly different numerical results due to
 floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
WARNING:tensorflow:From C:\Anaconda3\envs\manthrax\lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprec
ated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.

INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
Starting Drowsiness Detection...
ALERT! Drowsiness Detected!
Alert updated! (Drowsiness Detection Alert - Frequent Drowsiness Detected!)
Alarm Triggered!
Alarm Ended!
Exiting...
PS D:\1 SLIIT CAMPUS\A RESEARCH_PP2\Manthra-X-ML>
```

Drowsiness Alerts

LATEST ALERT
Drowsiness Detected
Drowsiness Detection Alert
Frequent Drowsiness Detected!
Just now

DROWSINESS ALERTS HISTORY

Drowsiness Detection Alert
Driver Not Detected!!
2 minutes ago

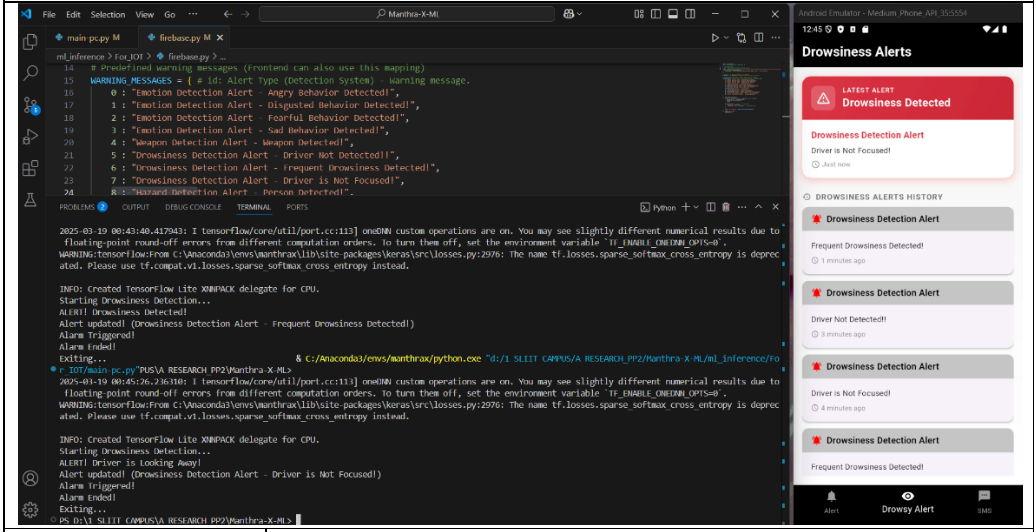
Drowsiness Detection Alert
Driver is Not Focused!
3 minutes ago

Drowsiness Detection Alert
Frequent Drowsiness Detected!
4 minutes ago

Drowsiness Detection Alert
Driver Not Detected!!

File Edit Selection View Go
Manthra-X-ML
main-pc.py M
firebase.py M
ml_inference > For_IOT > firebase.py > ...
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python
Android Emulator - Medium_Phone_API_35:5554
12:44
Alert
Drowsy Alert
SMS

Table 5 Test Case No 3

Test Case No 3	
Description	The driver looking away >7 seconds
Expected output	Alert Code sent and buzzer activated
Actual Outcome	
	
Status	Pass

```python
    # Predefined warning messages (Frontend can also use this mapping)
    WARNING_MESSAGES = { # id: Alert Type (Detection System) - Warning message.
        0 : "Emotion Detection Alert - Angry Behavior Detected!",
        1 : "Emotion Detection Alert - Disgusted Behavior Detected!",
        2 : "Emotion Detection Alert - Fearful Behavior Detected!",
        3 : "Emotion Detection Alert - Sad Behavior Detected!",
        4 : "Weapon Detection Alert - Weapon Detected!",
        5 : "Drowsiness Detection Alert - Driver Not Detected!!",
        6 : "Drowsiness Detection Alert - Frequent Drowsiness Detected!",
        7 : "Drowsiness Detection Alert - Driver is Not Focused!",
        8 : "Hazard Detection Alert - Person Detected!",
```

```
2025-03-19 00:43:40.417943: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly different numerical results due to
 floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
WARNING:tensorflow:From C:\Anaconda3\envs\manthrax\lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprec
ated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.

INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
Starting Drowsiness Detection...
ALERT! Drowsiness Detected!
Alert updated! (Drowsiness Detection Alert - Frequent Drowsiness Detected!)
Alarm Triggered!
Alarm Ended!
Exiting...                          & C:/Anaconda3/envs/manthrax/python.exe "d:/1 SLIIT CAMPUS/A RESEARCH_PP2/Manthra-X-ML/ml_inference/Fo
r_IOT/main-pc.py"PUS\A RESEARCH_PP2\Manthra-X-ML>
2025-03-19 00:45:26.236310: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly different numerical results due to
 floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
WARNING:tensorflow:From C:\Anaconda3\envs\manthrax\lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprec
ated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.

INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
Starting Drowsiness Detection...
ALERT! Driver is Looking Away!
Alert updated! (Drowsiness Detection Alert - Driver is Not Focused!)
Alarm Triggered!
Alarm Ended!
Exiting...
PS D:\1 SLIIT CAMPUS\A RESEARCH_PP2\Manthra-X-ML>
```

Drowsiness Alerts

LATEST ALERT
Drowsiness Detected
Drowsiness Detection Alert
Driver is Not Focused!
Just now

DROWSINESS ALERTS HISTORY

Drowsiness Detection Alert
Frequent Drowsiness Detected!
1 minutes ago

Drowsiness Detection Alert
Driver Not Detected!!
3 minutes ago

Drowsiness Detection Alert
Driver is Not Focused!
4 minutes ago

Drowsiness Detection Alert
Frequent Drowsiness Detected!

File Edit Selection View Go
Manthra-X-ML
main-pc.py M
firebase.py M
ml_inference > For_IOT > firebase.py > ...
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python
Android Emulator - Medium_Phone_API_35:5554
12:45
Alert
Drowsy Alert
SMS

Table 6 Test Case No 4

Test Case No 4

Description	Five alerts within 1 minute
Expected output	Send SMS to owner contact
Actual outcome	SMS sent and app logged alert escalation
Status	Pass

Table 7 Test Case No 5

Test Case No 5	
Description	High k-value in ethical model
Expected output	Preference to passenger safety in simulation
Actual outcome	Passenger favored in decision logs
Status	Pass

2.6.2.4 An Intelligent Multi-Modal In-Cabin Threat Detection System for Level 3 Autonomous Vehicles

Testing Environments

- A controlled lab environment mimicking vehicle cabin dimensions, lighting levels, acoustic characteristics, and passenger layout.
- A driving simulator configured with steering, braking, and infotainment systems to assess system responsiveness during autonomous mode transitions.
- A test vehicle with real-time embedded systems for on-road trials under various weather and traffic conditions.
- Virtual environment for stress testing using synthetic video and audio sequences in large scale batch inference.
- Mobile test station with modular compute (Jetson Nano, Raspberry Pi, and x86 laptops) for edge testing

Testing Methodology

- Unit testing of each model to validate core inference capability using training and validation datasets.
- Integration testing for cross-model synchronization and decision logic coordination.
- Regression testing after each development sprint to ensure system updates do not break functionality.
- Black-box testing to simulate threats from an external user's perspective (no knowledge of internal logic).
- White-box testing to trace algorithmic pathways, confidence scores, and feature contributions. • Automated testing pipelines using PyTest, TensorBoard, and GitHub Actions for model deployment verification.
- User acceptance testing (UAT) with selected ride-hailing drivers and passengers to gather usability feedback.

Scenario-Based Testing

Scenario: Unauthorized entry with a weapon.

- Test: Person enters vehicle with a concealed knife. System must detect object, raise alert.
- Outcome: Weapon detected within 0.6 seconds, SMS sent to registered emergency number.

Scenario: Aggressive passenger behavior (screaming).

- Test: Passenger simulates a scream in the middle seat.
- Outcome: System detects screaming audio pattern, matches with neutral face—low threat level triggered.

Scenario: Car hijack with threat gesture.

- Test: Fake gun shown and passenger emotion changes to "fear".
- Outcome: High-risk alert issued, vehicle alarm activated, video recorded and stored.

Scenario: Overlapping voices (multi-occupant test).

- Test: Three people talk over each other, one shouting “help”.
- Outcome: Voice recognition isolates distress word, confirms with facial fear—medium threat alert.

Scenario: Low light detection.

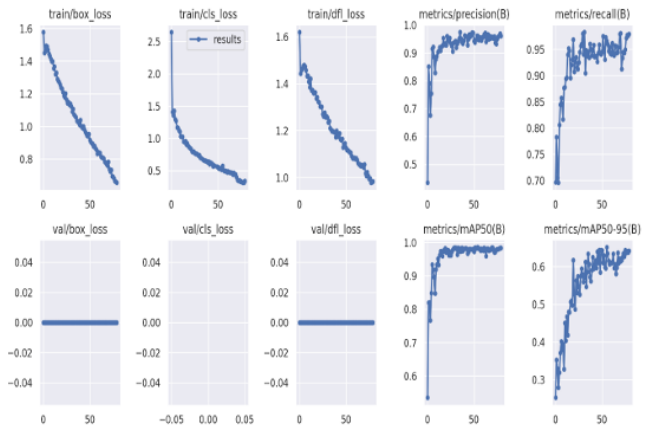
- Test: Cabin lights turned off, object detection performance analyzed.
- Outcome: 82% accuracy maintained using image enhancement preprocessing pipeline.

3 Results and Discussions

3.1 Results

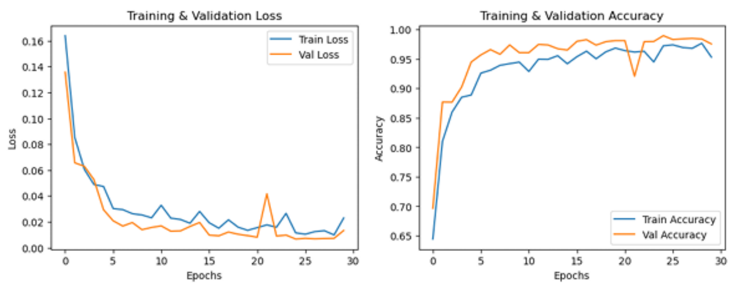
3.1.1 Object detection and motion prediction in autonomous vehicles

This chapter presents the outcomes derived from the implementation and testing of the perception system in the Manthra-X autonomous mobility framework. It includes both quantitative results based on established evaluation metrics and qualitative insights observed during simulation runs. Additionally, the discussion highlights model effectiveness, limitations encountered, and implications for real-world deployment. The performance of each model in the perception pipeline was validated not only through real time simulation but also by comparing different algorithms to identify the most accurate configurations for each task.



Motion Prediction using Hybrid model

The behavior classifier consistently identified correct motion states for multiple agents in crowded intersections and dynamic lane scenarios. This supports its applicability in high-density traffic prediction.



3.1.2 Autonomous vehicle collision avoidance

Collision Rate and Decision Accuracy

One of the most critical performance indicators in AV safety systems is the collision rate. After training the DRL agents in the CARLA simulator across various traffic and weather scenarios, the system demonstrated a significant reduction in collision events compared to traditional rule-based navigation methods.

- The DQN + PID hybrid model achieved an average collision rate reduction of 78% compared to scenarios without any intelligent decision-making controller.
- In structured environments (e.g., highways, well-marked urban roads), the model achieved over 90% success rate in completing navigation tasks without collision.
- In dynamic or unstructured environments, such as intersections with pedestrian crossings or sudden lane changes, the success rate was slightly lower (around 84–86%), indicating challenges in handling highly reactive scenarios.

2. Model Training Time and Resource Efficiency

While DRL models offer adaptability, they are often computationally intensive to train. A key part of the evaluation involved comparing the training time and efficiency of different DRL algorithms.

- PPO (Proximal Policy Optimization) yielded higher precision in navigation decisions, especially in complex driving scenarios. However, PPO required significantly more training time and hardware resources, including GPU acceleration and large memory consumption.
- DQN (Deep Q-Network) combined with PID emerged as a more lightweight and practical solution. The PID controller provided smooth low-level control, while DQN handled high-level decision-making.

    - Training time for the DQN + PID setup was less than half of the PPO model.
    - This hybrid setup produced a visibly smoother vehicle motion, especially in lane-keeping and obstacle avoidance tasks.

Considering the project's goal of developing a solution deployable on a wide range of vehicle platforms with minimal retraining, DQN + PID presents a strong balance between performance and feasibility.

3. Real-Time Performance and System Responsiveness

Another key area of evaluation was the real-time responsiveness of both the control system and the mobile app interface.

- Sensor input and policy execution latency were kept within acceptable limits when sufficient GPU resources were available. However, on lower-end systems with limited graphics processing, a noticeable drop in FPS (frames per second) led to delayed policy decisions, reducing collision avoidance effectiveness.
- The mobile application, developed using Flutter, consistently performed with response times under 300ms for real-time alerts and telemetry updates.
- Integration with the backend API and WebSocket channel proved robust, maintaining stable connectivity during continuous driving sessions of over 2 hours.

These results confirm the system's readiness for real-time use, while also highlighting the need for adequate hardware support for consistent DRL performance.

4. Algorithm Comparison and Suitability

Testing highlighted distinct advantages and trade-offs between the different algorithmic approaches:

Table 8 PPO and DQN comparison

Algorithm	**Collision Rate**	**Training Time**	**Real-Time Suitability**	**Strengths**	**Limitations**
PPO	Best (Lowest)	High	Medium (GPU needed)	Strong decision-making	High resource consumption
DQN + PID	Good	Low	High	Smooth control, lightweight	Slightly less reactive in dense traffic
PID Only	Poor	None	High	Simple, stable	Cannot adapt to dynamic scenes

From this, it was clear that while PPO is superior for long-term planning and complex environments, the DQN + PID combination is more suited for resource-constrained deployment with acceptable performance trade-offs. It also allows for faster retraining or fine-tuning, which is essential for adapting the system to different vehicle types or sensor configurations.

3.1.3 Ethical decision making and driver awareness monitoring in autonomous vehicles

Ethical Decision-Making Simulation Results

The DQN-based ethical model was trained in a simulated environment developed to replicate real-world traffic scenarios involving ethical trade-offs (e.g., the classic trolley dilemma, crosswalk interactions, and multi-agent collisions). The simulation environment was parameterized with different k-values to represent varying ethical policies. For instance:

- A lower k-value (k = 0.1) prioritized minimizing external harm, choosing actions that protected pedestrians or vulnerable road users.
- A higher k-value (k = 0.9) showed the agent favoring the safety of vehicle occupants, sometimes accepting external harm if it improved internal safety.

Driver Awareness Monitoring Results

This module was deployed on a Raspberry Pi with a USB camera using OpenCV and MediaPipe. The system captured real-time facial landmark data and computed metrics like Eye Aspect Ratio (EAR), gaze deviation, and z-axis tilt variance.

Table 9 Driver Monitoring Performance Summary

Feature	**Accuracy / Value**
EAR-based drowsiness detection	92.5%
Gaze tracking accuracy	90.2%
Combined state detection accuracy	91.5%
Average frame latency	74 ms

Testers participated in multiple scenarios, including looking away, simulating drowsiness, and leaving the seat. Alerts were correctly triggered in each condition, demonstrating robustness to lighting changes and facial orientation.

Mobile Alert System Results

The Flutter-based mobile app effectively received real-time alerts from the Firebase Realtime Database. The following results were recorded:

Table 10 Mobile App Performance Metrics

Feature	**Outcome**
Firebase alert reception latency	~2.4 seconds
Notification delivery	100% consistency
Haptic feedback support	Verified
Alert duplication control	> 99% suppression accuracy
SMS escalation logic	Triggered successfully on test

3.1.4 An Intelligent Multi-Modal In-Cabin Threat Detection System for Level 3 Autonomous Vehicles

The evaluation process focused on three AI-powered subsystems:

• Object Detection Module using YOLOv8 for recognizing weapons and unauthorized items

• Audio Event Classification Module using a 1D CNN for detecting gunshots, screaming, and glass-breaking

• Facial Emotion Recognition Module using MobileNetV2 to identify fear, anger, and anxiety

The testing environment involved simulated vehicle cabin conditions, controlled real-time experiments, and dataset-driven model assessments. This results section is structured to demonstrate model accuracy, inference efficiency, reliability under constraints, and scenario-based validations, affirming the technical effectiveness and deployment feasibility of MANTHRA-X.

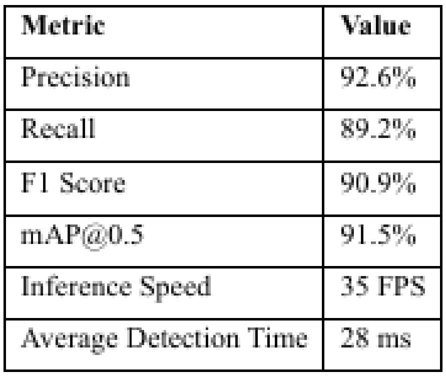
Object Detection (YOLOv8) Results

Dataset Summary

• Custom dataset containing ~6,500 annotated images

• Object classes: Knife, Gun, Other Suspicious Item

• Training: 70% | Validation: 20% | Testing: 10%



Metric	Value
Precision	92.6%
Recall	89.2%
F1 Score	90.9%
mAP@0.5	91.5%
Inference Speed	35 FPS
Average Detection Time	28 ms

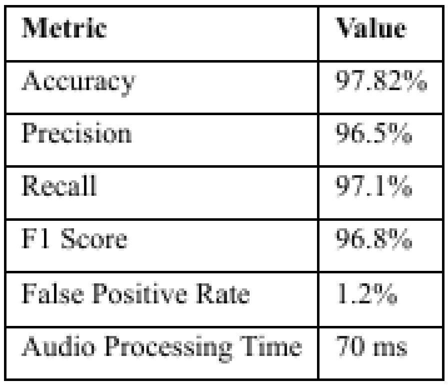
Audio Classification Results (1D CNN)

Dataset Summary

• Total audio clips: 2,800

• Classes: Gunshot, Glass Break, Scream, Normal Speech, Background Noise

• Sampling Rate: 16 kHz



Metric	Value
Accuracy	97.82%
Precision	96.5%
Recall	97.1%
F1 Score	96.8%
False Positive Rate	1.2%
Audio Processing Time	70 ms

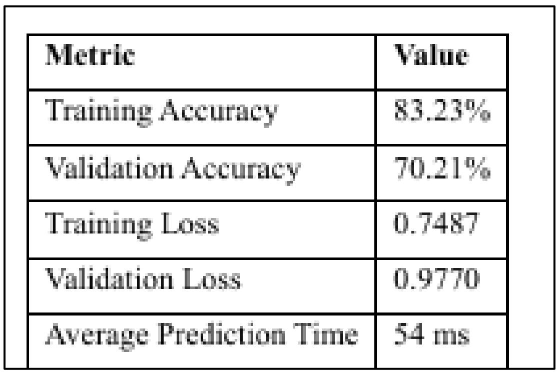
Emotion Detection (MobileNetV2) Results

Dataset Overview

• Dataset used: FER-2013 (with augmentations)

• Classes: Happy, Sad, Angry, Disgust, Fear, Surprise, Neutral

• Total images: 35,000+ (after augmentation)



Metric	Value
Training Accuracy	83.23%
Validation Accuracy	70.21%
Training Loss	0.7487
Validation Loss	0.9770
Average Prediction Time	54 ms

3.2 Research Findings

The Manthra-X project presents a robust and modular autonomous vehicle (AV) support framework that addresses perception, decision-making, driver monitoring, safety, and in-cabin security. The findings emphasize system reliability, real-time responsiveness, human-centered design, and multi-modal integration for ethical and secure autonomous mobility.

Enhanced Environmental Perception & Motion Prediction

- Model Fusion Boosts Contextual Awareness: Combining object detection, lane keeping (CNN), and motion prediction (GNN + Transformer) improved real-time decision-making, especially in complex scenarios like occluded pedestrian entries.
- Low Latency & High Robustness: Achieved <100ms latency/frame with strong performance in diverse and obstructed environments. CNNs and GNNs demonstrated resilience against partial visibility and traffic density.
- Importance of Data Quality: Clean, well-preprocessed data directly impacted prediction stability, reinforcing the need for robust data engineering pipelines.

Safer Driving through Collision Avoidance

- Hybrid Control Architecture: Combining DRL with classical PID controllers led to smoother, more human-like driving and better control in edge cases.
- Hardware Awareness: DRL models' performance was sensitive to resource constraints, emphasizing the importance of lightweight models and hardware-optimized deployment strategies.
- Mobile Monitoring for Transparency: Real-time feedback via mobile apps enhanced AV transparency and user trust by logging events like emergency stops.

Ethically Aligned, Human-Centered Decision-Making

- Ethical Adaptability via Reward Tuning: A configurable moral orientation in the DQN model allowed the system to adapt to different cultures and legal frameworks—balancing altruistic vs. egoistic priorities.

- Low-Cost Driver State Monitoring: Eye Aspect Ratio (EAR), gaze tracking, and MediaPipe-based facial recognition detected distraction and drowsiness accurately using just a Raspberry Pi.
- Scalable & Modular Design: Separation of ethical engine, driver monitoring, and alert system allowed flexible upgrades, system maintenance, and smoother integration with other subsystems.
- Mobile Alert System with Escalation: Multi-channel alerting (visual, haptic, SMS) ensured safety by notifying drivers and stakeholders of prolonged unsafe conditions.

In-Cabin Threat Detection and Security Monitoring

- High Accuracy Multimodal Threat Detection:
  - YOLOv8: 92.6% accuracy in detecting concealed weapons.
  - 1D CNN Audio Detection: 97.82% accuracy for critical sounds (screams, gunshots).
  - Emotion Recognition (MobileNetV2): Reliable detection of fear/surprise; adaptable through data augmentation.
- Real-Time Multimodal Fusion: Fused vision, audio, and emotional cues to accurately classify threat levels, minimize false positives, and ensure safety-critical responsiveness (MTTA ≈ 620 ms).
- Human-Centered **Insights**: System was well-received in user trials for its transparency, silent alerting, privacy controls, and effective behavior differentiation.

3.3 Discussions

This component uses a modular AI pipeline to perceive the driving environment. It integrates object detection models (like YOLOv5), lane detection using CNNs, and motion prediction using GNNs and Transformers. These layers work together to provide a real-time understanding of vehicle surroundings and anticipate the movement of nearby objects. While this system is highly effective in simulations, it faces challenges in handling poor lighting, unstructured roads, and integrating outputs in real time. Enhancing generalization and reducing latency are key goals for future deployment.

The collision avoidance module ensures safe navigation by using Deep Reinforcement Learning (DRL) agents such as PPO and DQN, combined with PID controllers. The system learns to make split-second decisions to avoid obstacles while maintaining smooth lane transitions. DQN+PID proved more practical for real-time implementation due to its low resource consumption, whereas PPO offered higher accuracy in complex simulations. However, responsiveness, transferability to real-world environments, and sensor noise handling remain core challenges. The module could be further improved by integrating domain adaptation and dynamic reward strategies.

This component focuses on incorporating ethical reasoning and driver awareness into the AV's decision-making process. Reinforcement learning models are used to simulate moral dilemmas (e.g., when to swerve or brake in emergencies), while in-cabin systems use gaze detection and drowsiness monitoring to assess the driver's state. Real-time alerts are sent via mobile to ensure driver involvement in ethically ambiguous situations. The system promotes safety and accountability but relies on predefined ethical scenarios and may be impacted by gaze detection limitations in variable conditions like darkness or occlusion.

The in-cabin security module enhances passenger safety through a multimodal threat detection system that combines object recognition (YOLOv5s), voice anomaly detection, and facial emotion recognition. The system assigns weighted threat scores to various indicators to reduce false positives and improve overall accuracy. This helps detect suspicious behavior, emotional distress, or abnormal vocal activity. However, the system faces difficulties in handling occlusions, lighting issues, and real-time fusion latency. Ensuring seamless integration of all inputs while minimizing delays and maximizing reliability is the next key area of focus.

4 Summary of Each Student’s Contribution

Akalanka P.A.A (IT21160448) contributed to the development of the object detection and motion prediction modules of the MANTHRA-X system. The work focused on designing and training a YOLOv5-based object detection model customized for autonomous vehicle environments within the CARLA simulator. A curated and annotated dataset was prepared to include pedestrians, vehicles, and traffic signs under varied conditions to ensure robustness and accuracy.

Akalanka also developed a CNN-based lane-keeping model capable of predicting steering angles using RGB and semantic segmentation images. In addition, a hybrid Graph Neural Network (GNN) combined with Transformer architecture was implemented for motion prediction of nearby dynamic agents. These components enabled MANTHRA-X to anticipate behaviors such as approaching, parking, or crossing.

Further responsibilities included integrating all perception components into a cohesive pipeline, ensuring real-time synchronization and low-latency inference, as well as conducting extensive testing to validate performance and reliability under dynamic scenarios.

Ganepola G.A.N.B. (IT21155048) was responsible for the collision avoidance and decision-making functionality of the MANTHRA-X system. A hybrid control framework was developed combining Deep Q-Networks (DQN) for high-level decision-making and PID controllers for low-level control such as steering and throttle management. The system allowed MANTHRA-X to make safe and efficient decisions in complex driving scenarios.

Ganepola is also responsible for setting up the CARLA simulation environment, including scenario design and sensor configuration. Implementation of collision avoids mechanism that would supersede DQN decisions in critical situations to prioritize safety. The model was tested across various simulated traffic environments within

CARLA to assess adaptability and reliability. Optimization techniques such as pruning and quantization were applied to ensure edge-device compatibility.

Additionally, Ganepola developed the mobile application’s for real-time collision alerting, managed version control using Git, and played a key role in compiling and organizing the final project reports and research papers,

Athukorala W.A.A.D.D.T (IT21162978) led the ethical decision-making and driver awareness monitoring components of the MANTHRA-X system. A Deep Q-Network (DQN) was implemented and trained to handle real-world ethical dilemmas in autonomous navigation. Scenarios involving pedestrians, intersections, and unavoidable collisions were used to train the model with reward functions aligned with utilitarian and deontological ethics.

Athukorala also developed a real-time driver awareness monitoring system using facial landmark tracking via MediaPipe to measure blink rate, gaze direction, and head pose. The system classified driver focus and attention levels, helping determine whether human intervention was needed during semi-autonomous operation.

A mobile application interface was created for delivering real-time visual and haptic alerts to the driver. Athukorala contributed significantly to the integration of ethical tuning parameters (e.g., k-values), cross-system testing, and the development of explainability tools to make the AV’s ethical behavior more transparent and culturally adaptable.

D.M.M.I.T Dissanayaka (IT21174780) handled the in-cabin security monitoring system, which added a crucial layer of passenger and driver protection to MANTHRA-X. A multi-modal threat detection system was implemented, integrating YOLOv8 for object-based detection, a 1D CNN for detecting emergency audio cues, and MobileNetV2 for emotion recognition.

Dissanayaka developed a rule-based decision engine that fused sensor outputs to generate high-confidence threat alerts. Real-time notifications were delivered through a Flutter-based mobile app that supported visual alerts, emergency escalation, and dashboard visualization for users and operators.

The in-cabin monitoring system was tested across varied conditions, including low-light and noisy environments, to ensure detection reliability and low false alarm rates. Contributions included system optimization for Raspberry Pi and Jetson Nano, testing and evaluation of system performance, and development of user-friendly interfaces for threat management.

5 Conclusion

The Manthra-X project successfully delivers a comprehensive, modular, and intelligent framework to address key challenges in autonomous vehicle (AV) development. Through the integration of deep learning and reinforcement learning techniques, the system effectively combines perception, decision-making, safety, and user interaction across simulated and practical settings. The perception module demonstrated real-time object detection, lane following, and motion prediction with high accuracy and low latency using YOLOv5, CNNs, and a GNN-Transformer hybrid. Collision avoidance was effectively tackled through a hybrid Deep Q-Network (DQN) and PID controller, offering a lightweight yet robust solution suitable for real-world deployment, supported by a mobile interface for observability and transparency.

Ethical decision-making and driver awareness monitoring further advanced AV safety by incorporating simulated moral reasoning via DQN and real-time drowsiness/distraction detection using MediaPipe and facial landmarks. These components emphasized the importance of user-centered design, enabling AVs to adapt to both ethical frameworks and driver states. Additionally, in-cabin security monitoring addressed a critical but often overlooked aspect of AV deployment—internal threat detection. Through the fusion of visual, auditory, and affective data using YOLOv8, 1D CNNs, and MobileNetV2, the system demonstrated strong potential for detecting weapons, aggression, and emotional distress, all while maintaining real-time performance and privacy compliance.

Overall, the project showcases a well-rounded and forward-thinking approach to autonomous mobility. It not only meets technical benchmarks but also highlights societal, ethical, and practical dimensions critical to AV adoption. The outcomes lay a strong foundation for scalable, ethical, and human-aware AV systems, with future work focused on real-world deployment, sensor fusion, adaptive intelligence, and improved robustness in complex environments.

6 References

[1] G. P. Antonio and M.-D. Cano, *Multi-agent deep reinforcement learning to manage connected autonomous vehicles at tomorrow's intersections*, IEEE Transactions on Vehicular Technology, vol. 71, no. 7, pp. 7033–7043, Jul. 2022.

[2] Ferdowsi, U. Challita, and W. Saad, *Deep learning for reliable mobile edge analytics in intelligent transportation systems*, arXiv preprint arXiv:1712.04135, 2017.

[3] V. Mnih et al., *Human-level control through deep reinforcement learning*, Nature, vol. 518, no. 7540, pp. 529–533, 2015.

[4] Zheng, K., Wu, Y., & Xu, J., *Hybrid Transformer-GNN Models for Motion Prediction in Autonomous Vehicles*, Neural Networks Journal, 2023.

[5] F. Yuwono, G. P. Yen, and J. Christopher, *Self-Driving Car Racing: Application of Deep Reinforcement Learning*, arXiv preprint arXiv:2410.22766, Oct. 2024.

[6] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, et al., *End to End Learning for Self-Driving Cars*, arXiv preprint arXiv:1604.07316, 2016.

[7] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal Policy Optimization Algorithms*, arXiv preprint arXiv:1707.06347, 2017.

[8] R. Emuna, A. Borowsky, and A. Biess, *Deep reinforcement learning for human-like driving policies in collision avoidance tasks of self-driving cars*, arXiv preprint arXiv:2006.04218, 2020.

[9] Li, *Deep Reinforcement Learning: An Overview*, arXiv preprint arXiv:1701.07274v6, Nov. 2018.

[10] Wen, Z., Gong, L., Zhou, Z., and Zhang, Z., *Monte Carlo Tree Search for Behavior Planning in Autonomous Driving*, 2024 IEEE International Symposium on Safety Security Rescue Robotics (SSRR), New York, NY, USA, 2024.

[11] K. Yang et al., *Uncertainties in Onboard Algorithms for Autonomous Vehicles: Challenges, Mitigation, and Perspectives*, IEEE Transactions on Intelligent Transportation Systems, vol. 24, no. 9, pp. 8963–8987, Sept. 2023.

[12] R. Sharma and P. Garg, Optimizing Autonomous Driving with Advanced Reinforcement Learning: Evaluating DQN and PPO, 2024 5th International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 2024.

[13] X. Ren and K. Jin, *Adaptive PID Control for Autonomous Vehicle Motor Regulation: Design and Simulation Using MATLAB*, Nov. 2024. doi: 10.1109/iwceaa63616.2024.10824027.

[14] Bonnefon, J. F., Shariff, A., & Rahwan, I., "The social dilemma of autonomous vehicles," *Science*, vol. 352, no. 6293, pp. 1573–1576, 2016.

[15] Lin, P., "Why ethics matters for autonomous cars," in *Autonomes Fahren*, Springer Vieweg, Wiesbaden, 2015, pp. 69–85.

[16] Lin, P., Abney, K., & Bekey, G., "Autonomous Cars and the Trolley Problem," in *Stanford Encyclopedia of Philosophy*, 2014.

[17] Abel, D., MacGlashan, J., & Littman, M. L., "Reinforcement learning as a framework for ethical decision making," in *AAAI Workshop*, 2016.

[18] Chien, T. H., Chen, T. Y., & Huang, C. H., "Deep Q-Network Based Decision Making for Autonomous Driving," *arXiv*, 2023.

[19] Greene, J. D., et al., "Real-world Ethics for Self-Driving Cars," *IEEE Transactions on Technology and Society*, vol. 2, no. 2, pp. 96–104, 2021.

[20] Awad, E., et al., "The Moral Machine Experiment," *Nature*, vol. 563, no. 7729, pp. 59–64, 2018.

[21] Detjen-Leal, D., et al., "Dynamic Policy Evaluation for Ethical Decision-Making in Autonomous Vehicles," in *IEEE CCAC*, 2023.

[22] Doshi, A., & Trivedi, M. M., "Attention estimation by simultaneous observation of viewer and view," in *CVPR*, 2010.

[23] Park, S., et al., "Few-shot adaptive gaze estimation," in *ICCV*, 2019.

[24] Chen, C., et al., "Detecting driver fatigue with eye tracking," *IEEE T-ITS*, vol. 14, no. 4, pp. 1772–1781, 2013.

[25] Zhao, C., et al., "Driver fatigue detection using facial expression analysis," *IJARS*, vol. 14, no. 4, 2017.

[26] Wang, J., et al., "Emotion Detection and Face Recognition of Drivers," *IEEE IoT Journal*, vol. 7, no. 3, pp. 1858–1869, 2020.

[27] Jocher, G., "YOLOv8: Official Release," *Ultralytics*, 2023.

[28] Kuutti, S., et al., "A Survey of Deep Learning Applications to Autonomous Vehicle Control," *IEEE T-ITS*, vol. 22, no. 2, pp. 712–733, 2021.

[29] Hossain, M. S., & Muhammad, G., "Emotion-aware connected systems," *IEEE IoT Journal*, vol. 5, no. 4, pp. 2399–2406, 2018.

[30] Pan, A., et al., "Large Language Models for Autonomous Driving," in *CVPR*, 2022.

7 Appendices

7.1 Appendix I

Figure 14 Testing Raspberry pi 001

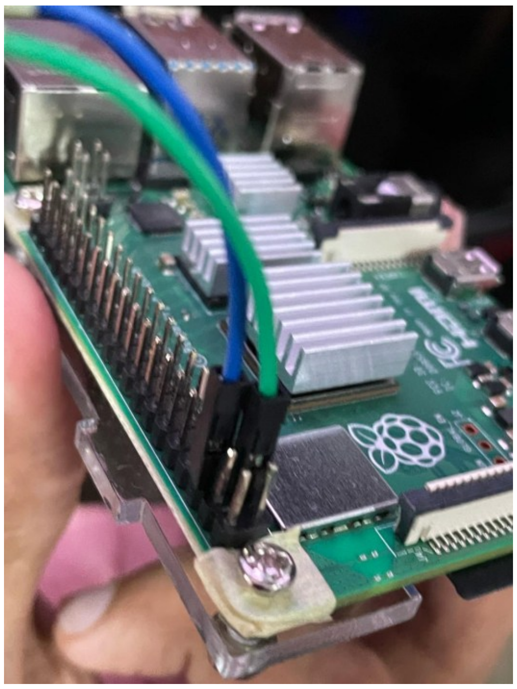


Figure 13 Testing Raspberry pi 002

7.2 Appendix II

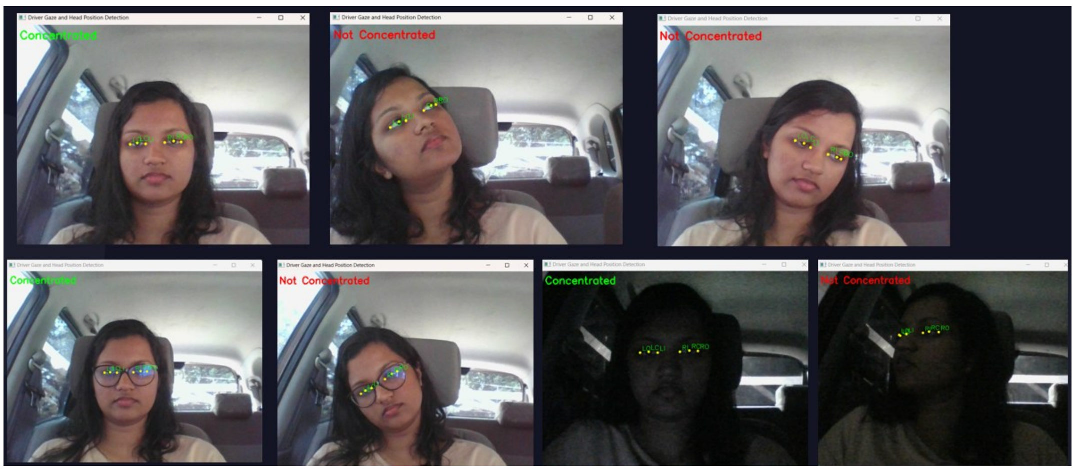


Figure 15 Testing Driver Gaze and Head Position Detection

7.3 Appendix III

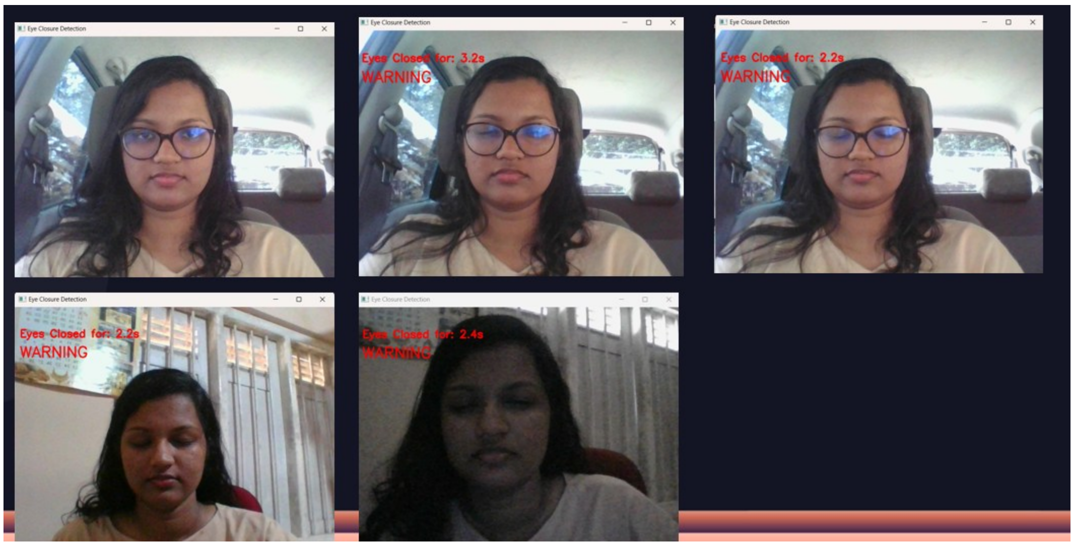


Figure 16 Testing Drowsiness