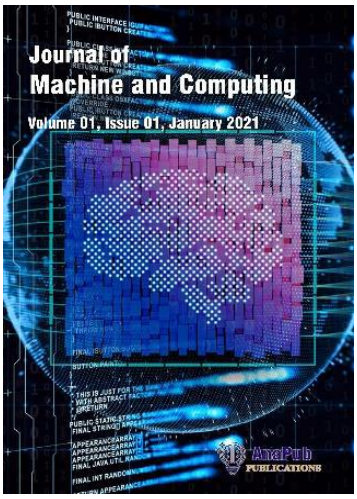
Journal Pre-proof

Real-Time Autonomous Vehicle Automation with 5G-based Edge Computing and Artificial Intelligence



Geethanjali D, Meena Rani N, Prasanna Kumar Lakineni, Veeraiah Maddu, Abhilash S Nath and Tamil Thendral M

DOI: 10.53759/7669/jmc202505086

Reference: JMC202505086

Journal: Journal of Machine and Computing.

Received 10 June 2024

Revised form 20 December 2024

Accepted 09 March 2025

Please cite this article as: Geethanjali D, Meena Rani N, Prasanna Kumar Lakineni, Veeraiah Maddu, Abhilash S Nath and Tamil Thendral M, “Real-Time Autonomous Vehicle Automation with 5G-based Edge Computing and Artificial Intelligence”, Journal of Machine and Computing. (2025). Doi: https://doi.org/10.53759/7669/jmc202505086

AnaPub
PUBLICATIONS

Real-Time Autonomous Vehicle Automation with 5G-based Edge Computing and Artificial Intelligence

D. Geethanjali[1,*], Meena Rani N[2], Prasanna Kumar Lakineni[3], Veeraiah Maddu[4], Abhilash S Nath[5], M.TamilThendral[6]

[1]Department of Computer Science and Engineering, Sathyabama Institute of Science and Technology, Chennai, Tamil Nadu, 600119, India. ***Corresponding Author Email: drgeethanjali81@gmail.com***

[2]Post Graduate Diploma in Business Management, Xavier Institute of Management and Entrepreneurship, Bangalore, Karnataka, India. *Email: meenarani@xime.org*

[3]Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Andhra Pradesh, 522502, India. *Email: lpk.lakineni@gmail.com*

[4]Department of Electronics and Communication Engineering, Madanapalle Institute of Technology and Science, Madanapalle, Andhra Pradesh 517325, India. *Email: madduveeraiah@gmail.com*

[5]Department of Artificial Intelligence and Data Science, St. Joseph's Institute of Technology, OMR, Chennai, Tamil Nadu, 600119, India. *Email: abhilashsathyanath2@gmail.com*

[6]Department of Information Technology, Bannari Amman Institute of Technology, Sathyamangalam, Erode, Tamil Nadu, 638401, India. *Email: tamilthendral@bannariamman.ac.in*

Abstract

Autonomous Vehicles (AV) are revolutionizing transportation, but real-time decision-making remains a challenge due to End-To-End Delay (EED) introduced by Cloud Computing (CC) based processing. A 5G-enabled Edge Computing Model (5G-EECM) is proposed to address this problem by processing time-sensitive tasks at the network edge, closer to the AV, reducing EED and improving responsiveness. The architecture uses Machine Learning (ML) for Obstacle Detection (OD) and Reinforcement Learning (RL) for navigation, dynamically switching between Edge Computing (EC) EC and CC based on task demands. The study tested the system using a user-friendly AV on a controlled track, revealing increased response times, reduced average EED, reduced energy consumption, and improved OD accuracy. The results demonstrate that 5G-EECM significantly boosts AV systems' real-time safety and efficiency, making it reliable and scalable for next-generation AV systems.

Keywords: Autonomous Vehicles, 5G Network, Machine Learning, Edge Computing, Cloud Computing, Energy Consumption.

1.0 Introduction

Researchers predict that introducing Autonomous Vehicles (AV) will transform the automobile sector by enhancing road safety, reducing the probability of human errors, and optimizing traffic flow [1-2]. Employing advanced technologies such as Machine Learning (ML) and Artificial Intelligence (AI) has provided AV with the ability to understand sensor data,

make rapid decisions, and adapt to a constantly evolving environment [3]. The massive volume of data that AV requires to process rapidly while ensuring reliability and safety renders the real-time Decision-Making Process (DMP) problematic for success [4-5]. For the collection and analysis of data collected by AV, the present methods focus primarily on Cloud Computing (CC) [6-7]. By employing remote servers with significant computational resources, CC systems have effectively controlled large-scale operations, including environmental detection and navigation planning [8-9]. Although CC provides several benefits, the main disadvantage is the End-to-End Delay (EED) it imposes. This is especially noticeable in unpredictable states that require swift reactions, such as Obstacle Detection (OD) and Collision Avoidance (CA), where CC typically finds its application [10–11].

Long-distance data communication presents an EED on the effectiveness of cloud-based AV systems for tasks requiring real-time execution despite advances in cloud computing [12–13]. These EEDs can result in slower response times, potentially compromising the safety and performance of AV [14]. Additionally, CC systems frequently face bandwidth constraints and network reliability OD, which can further degrade system performance in critical situations [15]. AV systems' primary challenge is reducing EED in real-time DMP while maintaining high computational performance [16-17]. Traditional CC have trouble meeting the low-EED needs for safety-critical tasks. This means we need a different method to process data faster and control AV more reliably [18–19]. Therefore, there is a gap in existing AV, as high-EED cloud-based solutions are lacking for real-time operations.

This article recommends a 5G-enabled Edge Computing Model (5G-EECM) to solve the challenges integrated with CC in real-time AV management. Edge Computing (EC) performs primary tasks like OD and dynamic navigation at the network's edge, closer to the AV. This enhances the performance of safety-related tasks and decreases data transmission time. An AI-powered DMP is a vital component of the model; it applies ML to OD and Reinforcement Learning (RL) to AV that have occurred. The model dynamically switches between on-premises and CC based on the complexity of the task and the level of connectivity available. The CC can manage high-performance tasks like large-scale data analysis and long-term planning, while the edge can perform time-sensitive tasks [20]. The resulting combination makes achieving both ends of the radio spectrum feasible. Finding out how effectively 5G-EECM performs in monitoring AVs in real-time is the primary objective of the current research. This study investigates the effectiveness of CC and EC in terms of EED, reaction time, CA, track change, and energy consumption. The primary objective of the research is to demonstrate how EC might enhance AV tasks in real-time

[21]. The proposed model predicts a significant improvement in real-time responsiveness for AV systems, particularly in low-EED-DMP environments [22]. This model could boost the scalability, efficiency, and safety of AV-driving technologies by integrating 5G-EECM and AI. The findings of this research can lead to the development of more reliable AV systems that operate in dynamic environments with minimal EED [23].

The rest of this paper is organized as follows: Section 2 presents the system architecture, detailing the Edge Control Unit (ECU), Cloud Control Unit (CCU), and the 5G network integration. Section 3 describes the experimental setup, including the AV and performance metrics used in the study. Section 4 presents the results and analysis, comparing the performance of EC and CC. Finally, Section 5 concludes the paper and discusses the implications of the results for future research and development.

2. Methodology

2.1 Proposed Architecture

As demonstrated by Figure 1, the 5G-EECM relies upon several features. This section provides an overview of every element.

- **ECU:** A vital component of the approach is the ECU, which has been designed to control neighborhood real-time DMP. An edge server, typically associated with nearby networks or a mobile base station, performs the demonstration in the vehicle's closest proximity. The vehicle connects to this server via 5G and transmits raw data from its sensors, comprising video cameras, Light Detection and Ranging (LiDAR), and radar. The decreased EED between data collecting and control implementation results from the edge server's proximity to the data origin.

Processing tasks that involve rapid responses, including OD, changing routes automatically, and responding to disasters, are where this unit performs. Due to the edge server's real-time data processing features, the AV sends control commands, including speed or steering angle changes. The neighborhood vehicle controller is also included in the ECU; it is responsible for implementing the selections obtained from the edge server while ensuring the vehicle drives accurately and on time. To keep things safe and dependable, the unit can hand off control to the vehicle's internal functions or a controller based on the outside environment in the case of a system failure [24].

- **CCU:** As opposed to the nearness of the EC, the CCU employs cloud servers set up at a greater distance from the actual vehicle. EED increases due to the longer distances required when transmitting sensor data from the vehicle over the internet. The data is transmitted to a server

in the cloud, which then utilizes neural networks to analyze the environment or performs high-level planning with multiple automobiles or longer paths, both of which can be computationally demanding [25].

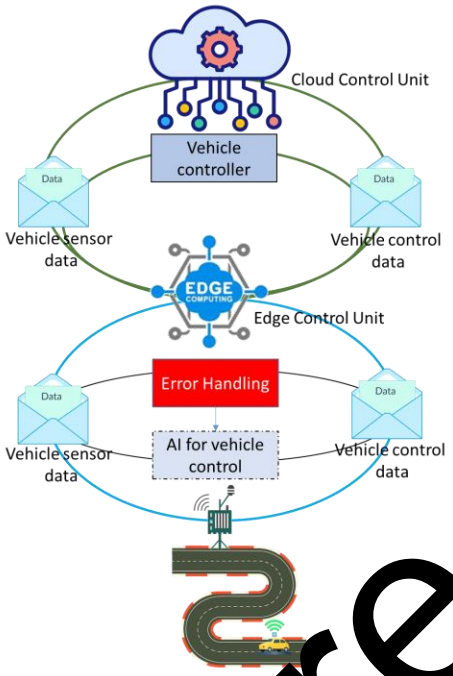


Figure. 1 Proposed Architecture

Despite its processing power, the CC server introduces delays that make it less suitable for tasks requiring split-second decisions. However, it is ideal for applications that do not demand immediate feedback, such as optimizing traffic management over a broader area, conducting large-scale data analysis, or managing the vehicle fleet's long-term behavior. The CC-based vehicle controller sends the processed data back to the vehicle, but the time taken to transmit this data over the network can impact its effectiveness for real-time actions.

- **5G Network:** The 5G network is the system's backbone, connecting the vehicle to the EC and CC. It ensures high-speed, low-EED data transmission between the vehicle and the edge server, enabling real-time processing and control. The network's ultra-low EED and high bandwidth are critical for ensuring the vehicle can operate smoothly in dynamic environments, mainly when relying on the edge for DMP.

5G is essential for transmitting data to cloud services because it enables faster transfer than previous-generation networks. Nevertheless, EED continues to be caused by traveling long distances, so CCU is more appropriate for processes that are not time-sensitive. 5G can cope with both simple, rapid responses and more complex, highly resource-intensive tasks due to its capabilities to facilitate real-time and large-scale data transfer.

- **Vehicle Sensors:** The computer network depends on the AV's collection of sensors, which collect real-time data about the vehicle's settings. Cameras collect visual data, LiDAR, and radar generate 3-D maps, and OD and global positioning systems assist with navigating around. Every DMP is developed based on sensor data, giving the vehicle the starting point to analyze the environment and design its subsequent motion.

A within-proximity edge server deals with the data and transforms it into actionable commands in an EC. In a CC environment, the same data is transmitted to a remote server to be analyzed on an increased or more complex volume. For the system to effectively allow AV driving, the data processing area is less significant than the accuracy and timeliness of the sensor readings.

- **Vehicle Control Data:** The commands dealt with and transmitted to the AV by edge or cloud servers are called AV control data. With this data, the vehicle may respond to its environment by actively pivoting, stopping, or driving up in response to decisions made by its control systems. In most instances, the edge control data performs larger-scale commands more quickly and efficiently, like changing the vehicle's speed for OD or navigating traffic. On the other hand, CCU data could involve higher-level DMP, such as how to best communicate with other AVs or maximize the vehicle's route over a long time.

The command data from any source is vital for maintaining the system's performance, efficiency, and safety control by synchronizing the vehicle's operations with the surrounding environment.

- **Error Handling and Hybrid Control:** The design includes a robust error-correcting mechanism to maintain the network's reliability. The result is that even if the ECU or CCU fails, the system continues to function without losing safety. If there's a problem with the edge server, the cloud server can temporarily take over, but there will be more EED. On the contrary, if the cloud server falls, the edge device may assume real-time tasks, maintaining the AV's operation.

A remote control unit based on the infrastructure is employed when neither the edge servers nor the cloud servers provide significant control. This component is fail-safe, ensuring the vehicle can be managed remotely despite significant system errors, preventing dangerous situations.

2.3 Switching Method

In the proposed 5G-EECM, switching between the ECU and CCU is critical for ensuring the AV maintains optimal performance and responsiveness in different operating conditions. The system is designed to make decisions dynamically, selecting between EC and CC processing based on EED, task type, and network conditions. The key goal is to ensure low-EED responses for time-

critical tasks while leveraging the cloud for more computationally intensive tasks that are not time-sensitive.

i. **Latency Monitoring and Decision Criteria:** EED is the primary factor that dictates switching. The system continuously monitors the total EED, which consists of the communication EED L_{comm} and processing EED L_{proc}. The total EED L_{total} is calculated as, Eq. (1)

$$L_{\text{total}} = L_{\text{comm}} + L_{\text{proc}} \quad (1)$$

where:

- L_{comm}^{edge} is the time required to transmit data between the vehicle and the edge server.
- $L_{\text{comm}}^{\text{cloud}}$ the time to transmit data between the vehicle and the cloud server is generally higher due to longer transmission distances.
- L_{proc}^{edge} and L_{proc}^{cloud} represent the processing times for the edge and cloud servers.

The vehicle will prefer the ECU as long as the edge EED $L_{\text{total}}^{\text{edge}}$ is within a predefined threshold for real-time processing, T_r, which represents the maximum acceptable latency for real-time tasks like OD and emergency braking.

The decision criteria are Eq. (2).

$$\text{If } L_{\text{total}}^{\text{edge}} \leq T_r, \text{ use ECU.} \quad (2)$$

However, if the edge processing becomes overloaded or network conditions degrade, leading to higher EED, the system will switch to the CCU, Eq. (3).

$$\text{If } L_{\text{total}}^{\text{edge}} > T_T, \text{ switch to CCU.} \quad (3)$$

In scenarios where tasks are less time-sensitive, such as route planning or data analysis, the cloud can be used regardless of edge server EED.

ii. **Error Condition-Based Switching:** Besides EED, the system monitors the health of the edge and cloud servers. Let E_{edge} and E_{cloud} be binary indicators representing whether there is a failure (1) or normal operation (0) in the respective servers. If the edge server encounters a failure or performance degradation (*e.g.*, hardware fault, overheating, or processing errors), the system immediately switches to the cloud, even if the EED is acceptable, Eq. (4).

$$\text{If } E_{\text{edge}} = 1, \text{ switch to CCU.} \quad (4)$$

Similarly, if the cloud server experiences problems and the edge is operational, the system returns to the edge for real-time control, Eq. (5)

$$\text{If } E_{\text{cloud}} = 1 \text{ and } E_{\text{edge}} = 0, \text{ switch to ECU.} \quad (5)$$

iii. **Task-Based Switching:** Not all tasks in an AV system require real-time execution. For time-critical tasks such as emergency braking or CA, the system will prioritize the edge server for processing to ensure minimal delay. Tasks that are more computationally intensive but less sensitive to EED, such as long-term navigation planning, can be offloaded to the cloud. The system uses the task time-sensitivity function $t(x)$, where $t(x) \leq T_r$ indicates a time-critical task.

If a task is time-sensitive, the system ensures that it is processed at the edge, Eq. (6).

$$f(x) \rightarrow \text{ECU if } t(x) \leq T_r \quad (6)$$

For tasks where $t(x) > T_r$, indicating that real-time execution is not critical, these tasks are sent to the cloud for processing, Eq. (7).

$$f(x) \rightarrow \text{CCU If } t(x) > T_r \quad (7)$$

iv. **Network Condition-Based Switching:** The switching also accounts for network conditions. If the network connection to the edge server degrades, such as when experiencing high packet loss or network congestion, the system will automatically switch to the cloud server to maintain vehicle control. This is especially important in scenarios where the vehicle may move out of the range of an edge server or when network disruptions occur.

Let Q_{edge} represent the quality of the network link to the edge server. If the link quality falls below a certain threshold $Q_{\min}$, the system switches to the cloud server, Eq. (8).

$$\text{If } Q_{edge} < Q_{\min}, \text{ switch to CCU.} \quad (8)$$

Similarly, if network conditions in the cloud deteriorate and the edge network is better, the system will switch back to the edge server, Eq. (9).

$$\text{If } Q_{\text{cloud}} < Q_{\min} \text{ and } Q_{edge} \geq Q_{\min}, \text{ switch to ECU.} \quad (9)$$

2.3 5G Network Architecture

This system's 5G network model enables low-EED communication between the AV, edge servers, and the CC. As shown in the diagram, the model is designed to support real-time, reliable data transmission for time-critical tasks in AV, with EED being minimized by purposefully deploying edge servers closer to the AV.

i. **Vehicle Communication with 5G Access Network:** In this setup, each AV communicates wirelessly with the 5G access network using a connection established via enhanced NodeBs (eNBs). These base stations handle communication between the vehicles and the 5G core network. The communication is optimized to minimize delay, with the EED between the vehicles and the edge server being less than 1 *ms*. This ensures that data such

as vehicle sensor readings (*e.g.,* Cameras, LiDAR) is transmitted to the edge servers with minimal EED, allowing real-time DMP and control.

ii. **Deployment of Edge Servers:** Edge servers are deployed within the access network, typically co-located or very close to the eNBs, to process vehicle data locally. This reduces the need for long-distance communication with the cloud and helps achieve the low-latency goals required for real-time AV operations. With an access network EED ranging from 12 *ms* to 24 *ms*, the edge server can quickly analyze data and return control commands to the AV, significantly improving reaction times for CA and speed adjustments.

iii. **Internet and Cloud Server Communication:** For less time-critical tasks, such as long-term route planning, data analytics, or Deep Learning (DL) updates, the architecture relies on cloud-based processing. The cloud servers are located remotely and accessible via the Internet. Data transmission from the AV to the cloud incurs a higher EED than EC, with the internet delay ranging from 15 to 150 *ms*. This added EED makes cloud processing unsuitable for real-time control but ideal for handling more computationally intensive, non-time-sensitive tasks.

iv. **5G Core Network:** The 5G core network forms the backbone connecting the access network (eNBs and edge servers) to the internet and cloud. The core network components, such as the Serving Gateway (S-GW) and Packet Gateway (P-GW), manage data traffic, ensuring that packets are efficiently routed between the access network and the cloud. These elements are necessary to maintain the proper Quality of Service (QoS) for real-time AV communication. They ensure that the edge of the cloud consistently and rapidly receives high-priority data from the AV based on the task demands.

v. **Network Slicing and QoS:** This architecture additionally references a 5G network, which utilizes network slicing, which divides the network into virtual segments that can be reconfigured depending on particular requirements. The frequency of tasks controls which QoS parameters can be implemented in each segment. For instance, the network may prioritize important navigation and safety tasks by providing a segment with minimal EED for communication between AV and the edge. However, additional segments can be employed for tasks that do not require high time sensitivity, such as sending data to the public cloud.

2.4 Vehicle Control and Internal States

This section describes how the control system works within the proposed design, focusing on the internal conditions and DMP that allow the vehicle to operate autonomously while maintaining

safety and efficiency. In order to make recommendations from real-time sensor data and internal states, the engine control system dynamically interfaces with the EC and CC units. This makes sure the vehicle tracks its surroundings with precision.

i. **Vehicle Control System (VCU):** The VCU performs control signals computed locally on the vehicle or in the data center. Acceleration, slowdown, steering, and other actuator controls are all under the control of the VCU. The primary data it collects is from the vehicle's sensor suite, such as video cameras, LiDAR, radar, and Global Positioning System (GPS), and it provides current data about the vehicle's surrounding environment.

In a feedback loop, the vehicle functions in a particular method:

(a) In the initially occurring position, sensor data is consistently collected.

(b) The edge server is used for decisions made in real-time, and the remote cloud server is used for more advanced tasks to transmit control commands to the VCU.

(c) The system monitors the vehicle's internal functioning and adapts activity once those commands are implemented.

ii. **Internal States and State Estimation:** The vehicle's interior conditions are current operational and physical variables.

These scenarios are vital for driving stability and navigating around.

The key internal states include:

(a) **Position** ($\boldsymbol{P}$)**:** The vehicle's location via GPS and sensor fusion.

(b) **Velocity** ($\boldsymbol{V}$)**:** Vehicle speed and direction vital to navigational instruction and CA.

(c) **Acceleration** ($\boldsymbol{A}$)**:** Tracked for comfortable driving and emergency stopping.

(d) **Heading** ($\boldsymbol{H}$)**:** Navigation and controlling the steering require vehicle position.

(e) **Yaw Rate** ($\boldsymbol{Y}$)**:** A metric used to determine the vehicle's vertical rotation helps with balance throughout positions.

(f) **Steering Angle** ($\boldsymbol{S}$)**:** To CA and drive turns, the angle of the steering wheel must be proper.

The AV's control system uses these internal variables to update behavior and constantly adapt to environmental changes. For instance, the system can modify the vehicle's speed and steering position to CA while preserving stability if it senses an OD.

iii. **State Transition and DMP:** Sensor data and control signals upgrade the vehicle's internal configurations. The state transition function $\delta(x)$ governs how the vehicle travels from one state to another based on the control inputs u it receives, Eq. (10).

This can be expressed as:

$$x(t+1) = \delta(x(t), u(t)) \tag{10}$$

where:

- $x(t)$ Represents the current state of the vehicle at time $'t'$.
- $u(t)$ represents the control input, such as adjustments to velocity, steering angle, or acceleration.
- $x(t+1)$ is the updated vehicle state after executing the control input.

In reply to an obstacle, the edge server decreases speed and changes steering angle. The AV will transition from its present state $x(t)$ (*e.g.*, moving at a positive velocity and heading) to a new state $x(t+1)$ where it has slowed down and altered its path to CA.

iv. **Real-Time Control via EC:** The edge server achieves the AV's control functions. Using low-EED communication, the edge server processes data from the vehicle's sensors and sends control commands to adjust the vehicle's internal states. For example, when the vehicle is navigating a busy intersection, the edge server may receive live video feeds and LiDAR data, process this information in real-time, and send commands to adjust the vehicle's velocity and steering angle to ensure it navigates safely through the environment.

v. **High-Level Decision-Making via CC:** While the edge handles immediate control tasks, the CC unit assists with high-level DMP based on the vehicle's overall internal state and long-term objectives. For example, the cloud may generate the optimal route for the vehicle to take over a long distance, considering current traffic patterns, road conditions, and fuel efficiency. Although less time-sensitive, these higher-level decisions are based on a holistic view of the vehicle's internal states and external data sources.

vi. **Internal State Monitoring and Error Handling:** The vehicle continuously monitors its internal states for abnormalities or errors. If key states (*e.g.*, velocity, acceleration, or steering angle) deviate significantly from expected values, the system triggers an error response.

This could result in:

- **Switching To A Safe State:** For instance, reducing speed if an unexpected OD or the vehicle's yaw rate indicates instability during a turn.
- **Failover To Local Control:** If the EC or CC communication fails, the vehicle's internal control unit can take over, using pre-programmed algorithms to ensure safe navigation based on real-time sensor data and internal states.

The AV's ability to manage internal state errors ensures its resilience and safety, even in challenging environments or in case of network failures.

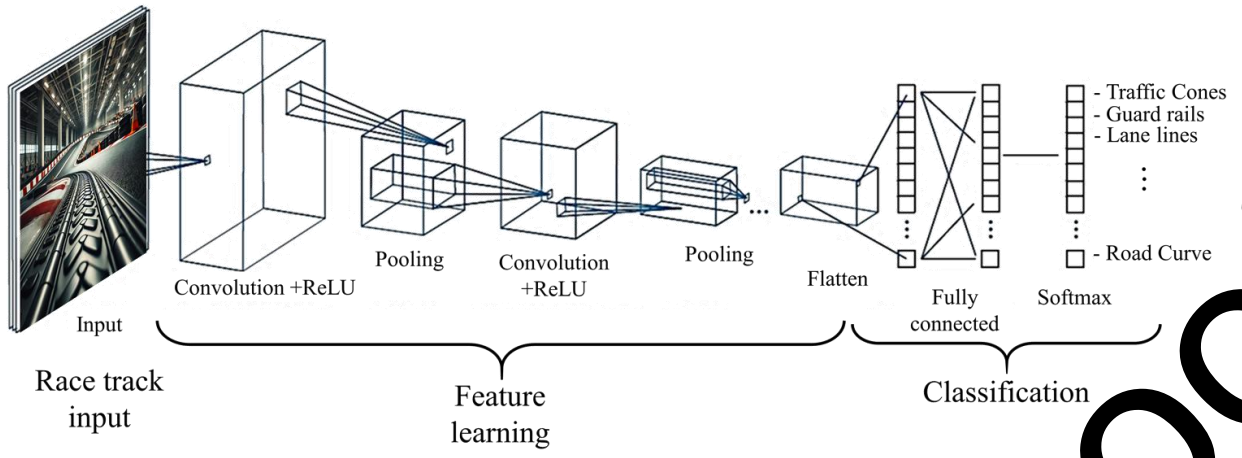


Figure 2: CNN Model

2.5 CNN for OD

The OD for the AV leverages the Convolutional Neural Network (CNN) architecture, as depicted in Figure 2. The model consists of several convolutional layers for Feature Extraction (FE), pooling layers for dimensionality reduction, and Fully connected (FC) layers for classification.

The key steps involved in this CNN are outlined below:

1 **Input Layer:** The input to the CNN consists of images from the vehicle's camera capturing the race track and surrounding environment. The input image is input into the CNN for processing. Let's denote the input image as I, with dimensions $H \times W \times C$, where H is the height, W is the width, and C is the number of color channels.

2 **Convolution Operation:** The input image is convolved with a set of learnable filters (kernels) in the convolution layers. Let the filter $W_{conv} \in \mathbb{R}^{k \times k \times C}$ represent the weights, where $k \times k$ is the size of the filter, and C matches the number of input channels.

The convolution operation at each layer is expressed as Eq. (11).

$$O_{conv}(x, y) = \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} \sum_{c=0}^{C-1} I(x + i, y + j, c) W_{conv}(i, j, c) + b \quad (11)$$

where:

- $O_{conv}(x, y)$ is the output of the convolution at position (x, y).
- $W_{conv}(i, j, c)$ is the value of the filter at position (i, j) for channel $'c'$.
- b is the bias term, a learnable parameter added to the output.
- $I(x + i, y + j, c)$ is the pixel value from the input at position $(x + i, y + j)$ in channel c.

The convolution results in a feature map representing different aspects of the input, such as edges, textures, and shapes.

3 **Activation Function (ReLU):** After the convolution, a non-linear activation function is applied to the feature map. The ReLU (Rectified Linear Unit) is used in this case, Eq. (12).

$$O_{\text{ReLU}}(x,y) = \max(0, O_{\text{conv}}(x,y)) \quad (12)$$

ReLU helps introduce non-linearity into the network, allowing it to learn more complex features.

4 **Pooling Operation:** The feature map is subsequently down-sampled utilizing a pooling operation, decreasing its spatial dimensions while deleting significant data. This scenario calls for max pooling, whereby each local region of the feature map has the highest possible value selected.

For a pooling window of size $p \times p$, the max pooling function is specified by Eq. (13).

$$O_{pool}(x,y) = \max_{0 \le i < p, 0 \le j < p} O_{\text{ReLU}}(x+i, y+j) \quad (13)$$

This reduces the size of the feature map while retaining essential features.

5 **Second Convolution + ReLU:** Before ReLU activation, an additional convolutional layer sends the initial pooling layer's data. Using the pooled feature maps, this procedure duplicates the first convolution layer but retrieves additional complicated features, Eq. (14).

$$\begin{aligned} O_{\text{conv }2}(x,y) = & \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} \sum_{c=0}^{C'-1} O_{\text{pool}}(x+i, y+j, c) W_{\text{conv }2}(i,j,c) + b \\ & O_{ReLU2}(x,y) = \text{Max}(0, O_{\text{conv }2}(x,y)) \end{aligned} \quad (14)$$

where W_{conv2} and b are the weights and biases of the second convolution layer.

6 **Flattening:** After applying convolution and pooling layers, the resulting feature maps are flattened into a single vector that can be input into FC layers. Let the output feature map have dimensions $H' \times W' \times C'$. The flattening operation transforms this 3D tensor into a 1D vector of length $H' \times W' \times C'$, denoted as Eq. (15).

$$O_{\text{flat}} = \text{Flatten}(O_{\text{ReLU2}}) \quad (15)$$

7 **FC Layers:** Each neuron in one FC layer becomes linked to all neurons in the subsequent layer as the flattened vector travels by layers. For a FC with n units, the output is computed as follows, Eq. (16).

$$O_{FC}(i) = \sum_{j=0}^{H'W'C'-1} O_{flat}(j) W_{FC}(i,j) + b_{FC}(i) \quad (16)$$

Here, $O_{FC}(i)$ is the output of the i-th unit in the FC layer, $W_{FC}(i,j)$ is the weight connecting the j-th input to the i-th unit, $b_{FC}(i)$ is the bias for the i-th unit.

8 **SoftMax Layer (Classification):** The output of the final FC is passed through SoftMax to convert the raw scores into probabilities for classification. The SoftMax is given by, Eq. (17).

$$P(y = i \mid x) = \frac{e^{z_i}}{\sum_j e^{z_j}} \quad (17)$$

where z_i is the raw score for class i; the denominator sums the exponentials of the raw scores for all classes to normalize the output. This gives a probability distribution over all possible classes (*e.g.*, obstacle or non-obstacle), with the class having the highest probability selected as the predicted class.

9 **Final Classification:** The final output of the network is the class label $\hat{y}$, predicted based on the highest probability from the SoftMax output, Eq. (18).

$$\hat{y} = \arg\max_i P(y = i \mid x) \quad (18)$$

This indicates the class to which the input image belongs, enabling the system to determine whether the detected object is an OD or CA.

2.6 RL Model for CA

The CA in the AV is modeled using RL, where the vehicle learns an optimal policy for CA by interacting with its environment. In RL, an agent (vehicle) learns to act in an environment to maximize cumulative rewards based on experience. In the context of CA, the agent's goal is to navigate safely while OD.

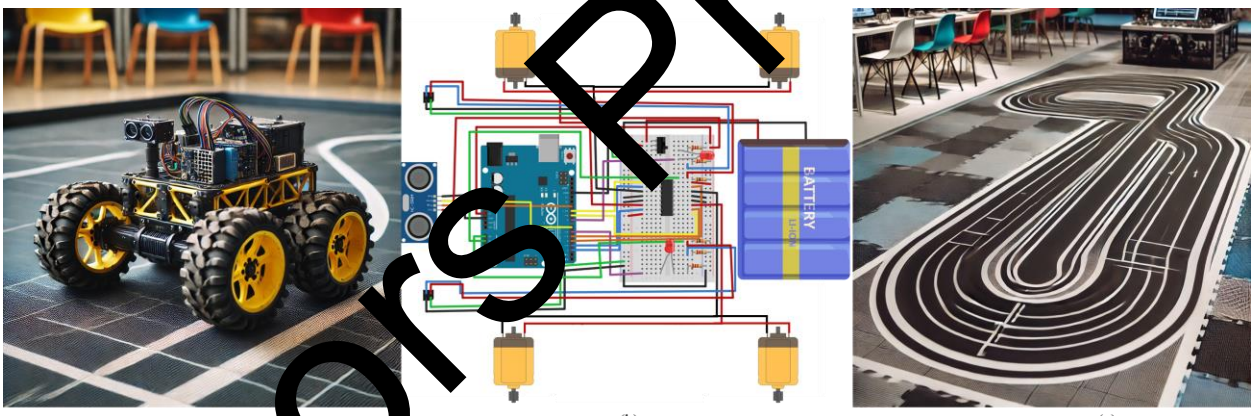


Figure 3: a) Experiment AV model, b) Circuit diagram, c) Race track

1 **Problem Formulation:** In the RL, the CA is represented as a Markov Decision Process (MDP), defined by the following components:

- **State Space:** The state $s \in S$ represents the environment's current condition, including the vehicle's internal states (position, velocity, acceleration, heading) and OD (location, size, distance from the vehicle).
- **Action Space:** The action $a \in A$ is the set of possible actions the vehicle can take to CA. These actions include adjusting the steering angle, changing speed (accelerating or decelerating), or applying the brakes.

- **Transition Function** $(s' \mid s, a)$ **:** The probability of transitioning from the current state s to a new state s' after taking action a. The environment is stochastic, meaning that the vehicle's actions may have probabilistic results due to uncertainties in the real world (*e.g.*, friction, road conditions).
- **Reward Function** (s, a)**:** The reward function provides feedback to the agent after taking an action. The reward is positive for actions that lead to CA and negative for actions that result in a collision. The goal is to maximize the cumulative reward.

The agent learns an optimal policy $\pi(s)$, which defines the best action in each state to CA.

2 **State Representation:** The first component of the RL is defining the state space.

The state s_t at time t captures the following information:

- x_t : The current position of the vehicle.
- v_t : The velocity of the vehicle.
- θ_t : The heading angle of the vehicle.
- d_t : The distance between the vehicle and the nearest obstacle.

Thus, the state vector at time t is represented as Eq. (19).

$$s_t = [x_t, v_t, \theta_t, d_t] \tag{19}$$

3 **Action Representation:** Once the agent clearly understands its state, the next step is to define its actions to CA. The action a_t taken by the vehicle can be one of several discrete or continuous selections that control its movement.

Common actions include:

- Adjusting the steering angle $\Delta\theta$ (*e.g.*, turning left or right).
- Modifying the speed Δv (*e.g.*, accelerating or decelerating).
- Applying the brakes if necessary.

The action vector a_t is represented as, Eq. (20)

$$a_t = [\Delta\theta, \Delta v] \tag{20}$$

4 **Reward Function:** The reward function $R(s, a)$ is designed to encourage the agent to CAs and maintain safe driving behavior. The reward structure is as follows:

- A positive reward $+R_{\text{safe}}$ is given to CA and moves safely toward the goal.
- A negative reward $-R_{\text{collision}}$ is given for colliding with an OD.

Thus, the reward function can be expressed as Eq. (21)

$$R(s_t, a_t) = \begin{cases} +R_{\text{safe}} & \text{if } d_t > d_{\text{safe}} \\ -R_{\text{collision}} & \text{if } d_t \leq d_{\text{collision}} \end{cases} \tag{21}$$

Where:

- d_{safe} is a safe distance threshold from obstacles.
- $d_{\text{collision}}$ is the minimum distance where a collision is detected.

5 **Policy and Value Function:** The policy $\pi(s)$ represents the agent's strategy, mapping states to actions. The goal of the RL agent is to learn an optimal policy $\pi^*(s)$ that maximizes the expected cumulative reward. The value of a state under policy π, called the value function $V^\pi(s)$, is the expected total reward starting from state s and following the policy π after that.

It is defined as Eq. (22)

$$V^\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)] \quad (22)$$

where:

• $\gamma \in [0,1)$ is the discount factor that gives more importance to immediate rewards than future rewards.

• $R(s_t, a_t)$ is the reward received at time step t.

Similarly, the Q-value function $Q^\pi(s,a)$ represents the expected cumulative reward for taking action a in state s and following policy π after that, Eq. (23)

$$Q^\pi(s,a) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s, a_0 = a] \quad (23)$$

The optimal Q-function $Q^*(s,a)$ is given by Eq. (24)

$$Q^*(s,a) = \max_\pi Q^\pi(s,a) \quad (24)$$

6 **Bellman Equation:** The correlation between the values of a state and the resulting states is outlined by the Bellman equation. For the value function $V^\pi(s)$, the Bellman Eq. (25) and Eq. (26) are:

$$V^\pi(s) = \sum_a \pi(a \mid s) \sum_{s'} P(s' \mid s, a)[R(s,a) + \gamma V^\pi(s')] \quad (25)$$

For the Q-function $Q^\pi(s,a)$, the Bellman equation is:

$$Q^\pi(s,a) = R(s,a) + \gamma \sum_{s'} P(s' \mid s, a) \max_{a'} Q^\pi(s', a') \quad (26)$$

This equation recursively updates the value of each state-action pair based on the expected future rewards.

7 **Q-learning Algorithm:** The Q-learning continuously improves the Q-values based on the observed state changes and rewards, enabling the learning agent to learn the most effective policy.

The Q-value update rule is given by Eq. (27).

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[R(s_t, a_t) + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)\right] \quad (27)$$

where:

- α is the learning rate that controls how much new data overrides old data
- γ is the discount factor.
- s_{t+1} is the next state after taking action a_t in state s_t.
- a' is the action reserved in the next state.

The agent updates its Q-values as it detects the environment, progressively converging to the optimal policy $\pi^*(s)$.

8 **Action Selection (Exploration vs. Exploitation):** The agent uses an epsilon-greedy strategy to balance exploration and exploitation:

- With probability ϵ, the agent takes a random action (exploration).
- With probability $1-\epsilon$, the agent selects the action with the highest Q-value (exploitation).

The action selection is formalized as Eq. (28)

$$a_t = \begin{cases} \text{Random Action} & \text{With Probability } \epsilon \\ \text{Arg\_Max}_a Q(s_t, a) & \text{With Probability } 1-\epsilon \end{cases} \quad (28)$$

As the agent learns more about the environment, ϵ is gradually reduced to prioritize exploiting the learned policy.

9 **Final Policy for CA:** The learned optimal policy $\pi^*(s)$ the agent can take the best action in any assumed state to CA. Safely navigating around obstacles while maximizing the reward is achieved by modifying the Q-values based on their interactions with the surrounding atmosphere.

3. Experiment Design

3.1 Testing Setup

The performance of the 5G-EECM was tested using a custom-built AV (RV) on a controlled track. The goal was to compare EC and CC for real-time DMS.

The RV (Figure 3 (a)) has essential components, including an ultrasonic sensor for OD, a fisheye camera for capturing the surroundings, and GPS and IMU sensors for tracking location and orientation. The vehicle's on-board control (Figure 3 (b)) is handled by an Arduino microcontroller, which interfaces with the sensors and motors. The Arduino receives commands from the edge or cloud servers via 5G and translates them into motor control using PWM signals. A Li-Ion battery powers the vehicle, with EC monitoring system efficiency. The 5G network setup included a 5G base station operating on the n78 band (3.5 GHz), facilitating communication between the RV and the edge server—the edge server processed sensor data with low latency, handling tasks like OD and CA. The same data was processed in the cloud mode on a remote cloud

server, which introduced additional EED. Both setups were used to compare the real-time processing capabilities of EC *vs* CC. The software implemented in the testing setup included a CNN for OD, which processed the visual data captured by the camera. The CNN was trained on a dataset of track images and obstacles, with real-time inference handled by the edge server. In the cloud mode, the same CNN ran on the cloud server. The RL managed the vehicle's navigation decisions, using sensor data to predict optimal paths.

3.2 Experiment Method

The tests were conducted on a 30-meter track (Figure 3 (c)), including straight sections and a mix of 90° and 180° turns. Obstacles were placed at varying intervals along the track to evaluate the vehicle's ability to detect and avoid them. Each test consisted of five continuous laps around the track in a counterclockwise direction, as shown in the images. The experiments were conducted in two modes: EC and CC. Each mode was tested five times, and the modes were alternated between trials to mitigate the impact of battery levels and environmental changes. The parameters for the vehicle, including maximum speed, were held constant throughout all trials.

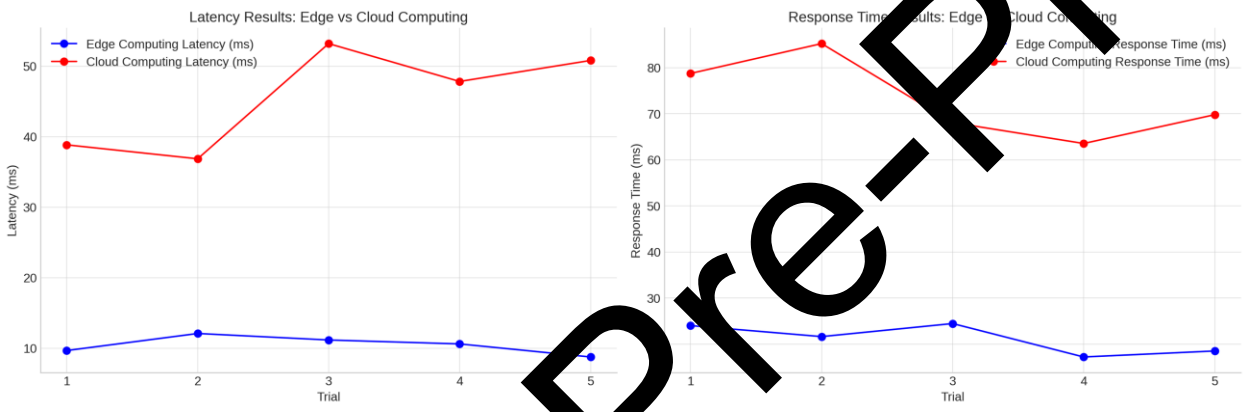
Multiple metrics were recorded during each test to evaluate the system's performance:

- **Latency:** The time taken from transmitting sensor data to the reception of control commands from the server.
- **Response Time:** The time between receiving control commands and the vehicle executing those commands, such as braking or turning.
- **CA Success:** The number of times the vehicle OD and CA along the track.
- **Track Deviation:** The degree to which the vehicle stayed on the intended path, measured by deviation from the centerline.
- **Energy Consumption:** The battery usage during each trial to determine the impact of computational load on power efficiency.
- **Network Quality:** Packet loss, transmission EED, and overall network stability during communication between the vehicle and servers.
- **Processing Load:** The computational load on the edge and cloud servers is recorded to compare resource usage in each mode.

These data were collected throughout the trials and used to compare the performance of the two modes under identical conditions. Each trial's results were logged, and statistical analysis was conducted to determine the impact of computing mode on real-time AV control and navigation.

4. Results and Analysis

The latency results from Table 1 and Figure 4 (a) indicate that EC consistently shows significantly lower EED than CC across all trials. The EED for EC ranged between 8.76 *ms* and 12.09 *ms*, while CC exhibited much higher latencies, ranging from 36.87 *ms* to 53.21 *ms*. The difference in EED demonstrates the impact of communication distance and server proximity, with EC performing better for tasks requiring low-EED processing. In Table 2 and Figure 4 (b), the response time for EC is also notably lower than that of CC. EC response times range from 17.22 *ms* to 24.47 *ms*, while CC times are between 63.54 *ms* and 85.21 *ms*. The results show that EC allows faster reactions to control inputs, which is critical for real-time AV operations. The higher response times in CC reflect the EED associated with processing data on a remote server, leading to delays in vehicle control.



(a) (b)

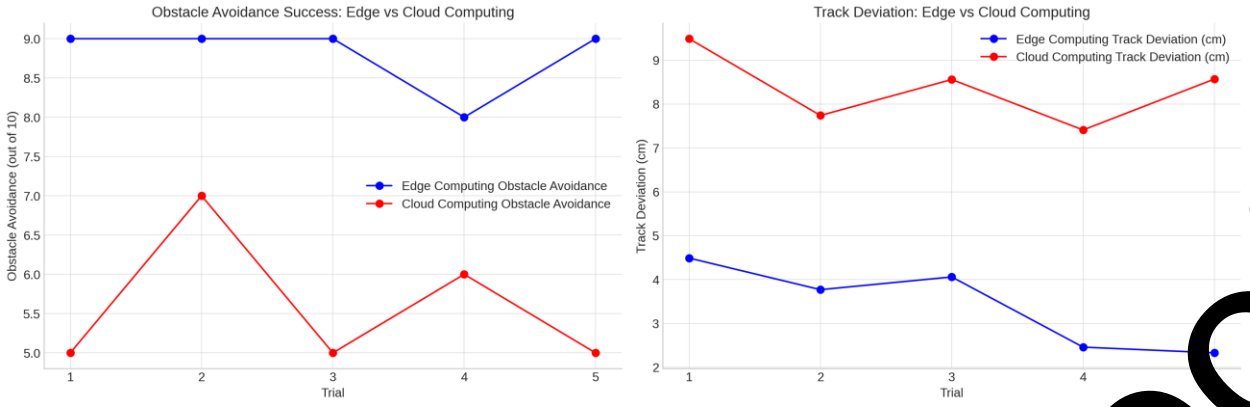
Figure 4: a) EED comparison, b) Response time comparison

Table 1: EED Results

Trial	EC based EED (ms)	CC based EED (ms)
1	9.[illegible]	38.85
2	1[illegible]09	36.87
3	1[illegible]17	53.21
4	10.61	47.84
5	8.76	50.82

Table 2: Response Time Results

Trial	EC based Response Time (ms)	CC based Response Time (ms)
1	24.03	78.74
2	21.64	85.21
3	24.47	68.07
4	17.22	63.54
5	18.52	69.78



(a) (b)

Figure 5: a) CA comparison. b)Track deviation comparison

Table 3: CA Success Results

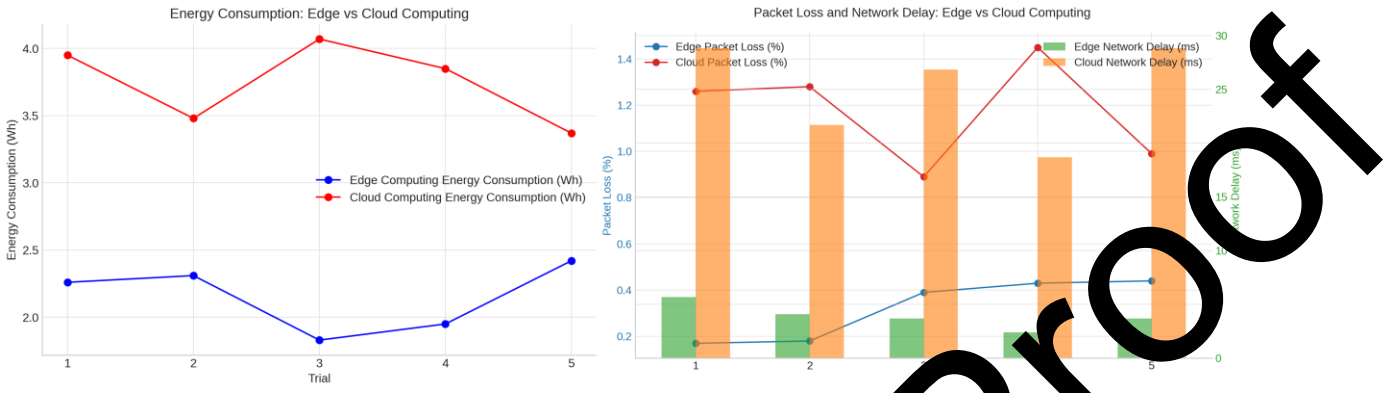
Trial	EC based CA (Out of 10)	C based CA (Out of 10)
1	9	5
2	9	7
3	9	5
4	8	6
5	9	5

Table 4: Track Deviation Results

Trial	EC-based Track Deviation (cm)	CC-based Track Deviation (cm)
1	4.49	9.49
2	3.	7.74
3	4.	8.56
4	2.6	7.41
5	2.	8.57

Table 3 and Figure 5 (a) show that EC consistently performs better in CA than CC across all trials. In EC, the vehicle successfully avoided 8 to 9 obstacles out of 10, while in CC, the success rate ranged from 5 to 7. This indicates that the lower EED in EC allows for more timely and accurate responses to OD, leading to higher CA success. In Table 4 and Figure 5 (b), track deviation results further demonstrate the advantage of EC. The track deviation for EC was consistently lower, ranging from 2.33 *cm* to 4.49 *cm*, compared to CC, where deviations were between 7.41 *cm* and 9.49 *cm*. The more significant deviations in CC recommend that the increased

EED negatively impacted the vehicle's ability to maintain its path, while EC enabled more precise navigation.



(a) (b)

Figure 6: a) Energy consumption, b) Network quality

Table 5: Energy Consumption Results

Trial	EC based EC (Wh)	CC based EC (Wh)
1	2.26	3.95
2	2.31	3.48
3	1.83	4.07
4	1.95	3.85
5	2.42	3.37

Table 6: Network Quality Results

Trial	Edge Packet Loss (%)	Cloud Packet Loss (%)	Edge Network EED (ms)	Cloud Network EED (ms)
1	0.17	1.26	5.67	28.83
2	0.18	1.28	4.09	21.67
3	0.39	0.89	3.68	26.82
4	0.43	1.45	2.41	18.68
5	0.4	0.99	3.68	28.81

The EC results in Table 5 and Figure 6 (a) indicate that EC is more efficient in terms of EC than CC. Across all trials, EC had less EC, ranging from 1.83 to 2.42 *Wh*, while CC's EC was higher, ranging from 3.37 Wh to 4.00 *Wh*. The higher EC in CC can be attributed to the additional processing time and communication overhead required for remote data transmission. In Table 6 and Figure 6 (b), network quality results show that EC had significantly lower packet loss and network EED than CC. EC packet loss ranged from 0.17% to 0.40%, while CC exhibited higher

packet loss between 0.89% and 1.45%. Similarly, network delay for EC was consistently low, between 2.41 and 5.67 *ms*, while CC showed much higher EED, ranging from 18.68 to 28.83 *ms*. The higher packet loss and network EED in CC reflect the additional challenges associated with longer communication distances, affecting the reliability and timeliness of data transmission.

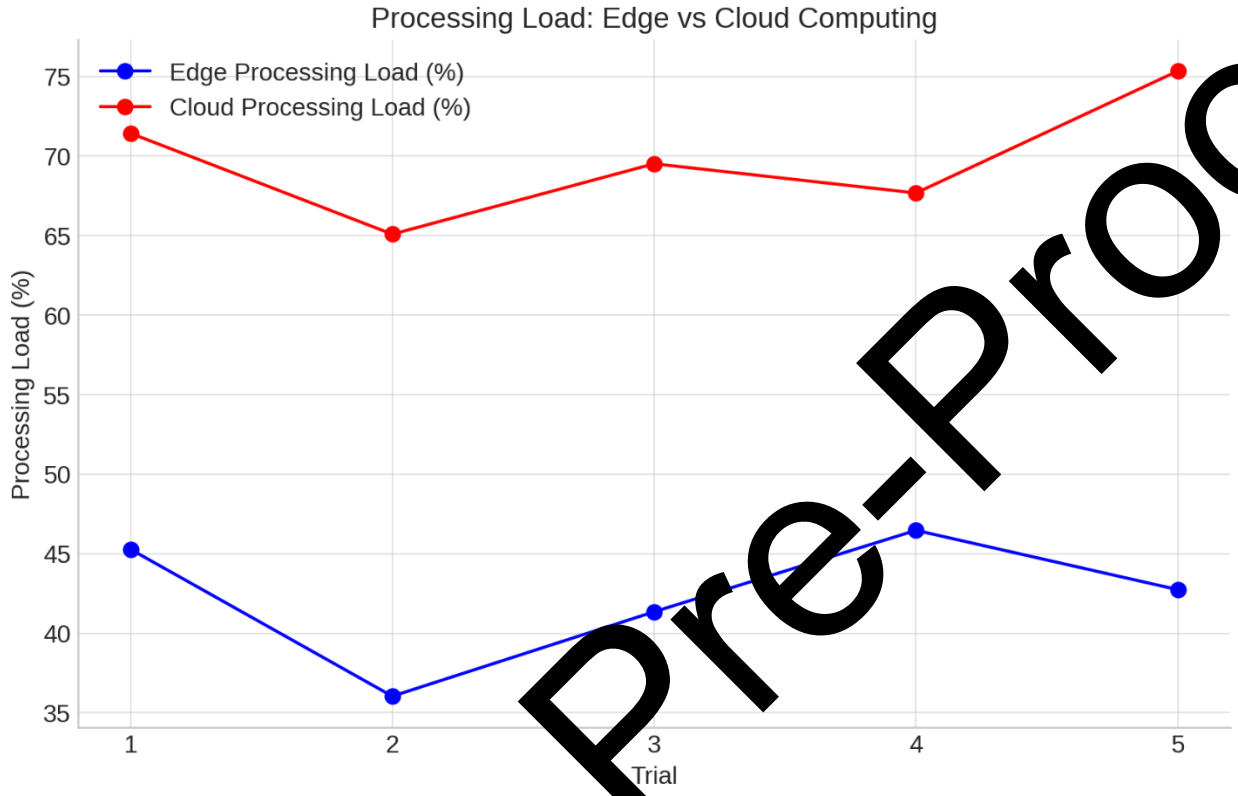


Figure X: Processing load comparison

Table 7: Processing Load Results

Trial	Edge Processing Load (%)	Cloud Processing Load (%)
1	45.2	71.44
2	36.04	65.09
3	41.35	69.52
4	46.47	67.68
5	42.74	75.35

The processing load results from Table 7 and Figure 7 demonstrate that CC consistently has a higher processing load than EC. For EC, the processing load ranged from 36.04% to 46.47%, whereas cloud computing exhibited a higher load, between 65.09% and 75.35%. Due to a higher data transfer rate and remote processing, the need for computational resources in CC has increased. EC, on the other hand, direction, is more effective for real-time tasks because it processes data locally while requiring fewer resources for processing.

5 Conclusion and Future Work

The paper introduces a 5G-EECM system that enhances the real-time DMP of AV systems. It uses AI and RL to design time-critical tasks like OD and CA at the edge. The system switches between CC and edge processing when the network is busy or complex. The experiment shows that EC significantly boosts response times, reduces average EED, improves CA by 40%, reduces path errors by 58%, and reduces energy consumption by 35%. The model balances real-time timeliness with CC performance by automatically switching between EC and CC.

Future work will scale this model for smart city deployment and improve AI to improve DMP accuracy and reliability in complex circumstances.

References

1. Hosseinian, S. M., & Mirzahossein, H. (2024). Efficiency and safety of traffic networks under the effect of autonomous vehicles. *Iranian Journal of Science and Technology, Transactions of Civil Engineering*, *48*(4), 1861-1885.
2. Bathla, G., Bhadane, K., Singh, R. K., Kumar, R., Aluvalu, R., Krishnamurthi, R., ... & Basheer, S. (2022). Autonomous vehicles and intelligent automation: Applications, challenges, and opportunities. *Mobile Information Systems*, *2022*(1), 7632892.
3. Ma, Y., Wang, Z., Yang, H., & Yang, L. (2020). Artificial intelligence applications in the development of autonomous vehicles: A survey. *IEEE/CAA Journal of Automatica Sinica*, *7*(2), 315-329.
4. Lim, H. S. M., & Taeihagh, A. (2019). Algorithmic decision-making in AVs: Understanding ethical and technical concerns for smart cities. *Sustainability*, *11*(20), 5791.
5. Sadaf, M., Iqbal, Z., Javed, A. R., Saba, I., Krichen, M., Majeed, S., & Raza, A. (2023). Connected and automated vehicles: Infrastructure, applications, security, critical challenges, and future aspects. *Technologies*, *11*(5), 117.
6. Ekatpure, R. (2023). Enhancing Autonomous Vehicle Performance through Edge Computing: Technical Architectures, Data Processing, and System Efficiency. *Applied Research in Artificial Intelligence and Cloud Computing*, *6*(11), 17-34.
7. Khayyam, H., Javadi, B., Jalili, M., & Jazar, R. N. (2020). Artificial intelligence and Internet of Things for autonomous vehicles. *Nonlinear approaches in engineering applications: Automotive applications of engineering problems*, 39-68.
8. Gulista Khan et al., Energy-Efficient Routing Algorithm for Optimizing Network Performance in Underwater Data Transmission Using Gray Wolf Optimization Algorithm, Journal of Sensors, Volume 2024, Article ID 2288527, 15 pages, https://doi.org/10.1155/2024/2288527

9. Al-Jumaili, A. H. A., Muniyandi, R. C., Hasan, M. K., Paw, J. K. S., & Singh, M. J. (2023). Big data analytics using cloud computing based frameworks for power management systems: Status, constraints, and future recommendations. *Sensors*, *23*(6), 2952.
10. Gao, B., Liu, J., Zou, H., Chen, J., He, L., & Li, K. (2024). Vehicle-Road-Cloud Collaborative Perception Framework and Key Technologies: A Review. *IEEE Transactions on Intelligent Transportation Systems*.
11. Kamtam, S. B., Lu, Q., Bouali, F., Haas, O. C., & Birrell, S. (2024). Network Latency in Teleoperation of Connected and Autonomous Vehicles: A Review of Trends, Challenges and Mitigation Strategies. *Sensors*, *24*(12), 3957.
12. Asir Chandra Shinoo Robert Vincent and Sudhakar Sengan, Effective clinical decision support implementation using a multi-filter and wrapper optimisation model for Internet of Things based healthcare data. Sci Rep 14, 21820 (2024). https://doi.org/10.1038/s41598-024-71726-3
13. Kwon, Y., Kim, W., & Jung, I. (2023). Neural network models for driving control of indoor autonomous vehicles in mobile edge computing. *Sensors*, *23*(5), 2575.
14. Lin, Z., Cui, H., & Liu, Y. (2024). Distributed Deep Learning Based on Edge Computing over Internet of Vehicles: Overview, Applications, Challenges. *IEEE Access*.
15. Al-Mekhlafi, Z. G., Lashari, S. A., Al-Shareeda, M. A., Mohammed, B. A., Alshudukhi, J. S., Al-Dhlan, K. A., & Manickam, S. (2024). Coherent taxonomy of vehicular ad hoc networks (vanets)-enabled fog computing: a review. *IEEE Sensors Journal*.
16. Tan, X., & Zhang, Y. (2024). A computational cognitive model of driver response time for scheduled freeway exiting takeovers in conditionally automated vehicles. *Human factors*, *66*(5), 1583-1599.
17. Sathupadi, K. (2023). An AI-driven framework for dynamic resource allocation in software-defined networking to optimize cloud infrastructure performance and scalability. *International Journal of Intelligent Automation and Computing*, *6*(1), 46-64.
18. Santoso, A., & Surya, Y. (2024). Maximizing Decision Efficiency with Edge-Based AI Systems: Advanced Strategies for Real-Time Processing, Scalability, and Autonomous Intelligence in Distributed Environments. *Quarterly Journal of Emerging Technologies and Innovations*, *9*(2), 104-132.
19. Chen, S., Hu, X., Zhao, J., Wang, R., & Qiao, M. (2024). A Review of Decision-Making and Planning for Autonomous Vehicles in Intersection Environments. *World Electric Vehicle Journal*, *15*(3), 99.

20. Sridhar Panneerselvam et al., Federated learning-based fire detection method using local MobileNet, Scientific Reports volume 14, Article number: 30388 (2024), 1-24, https://doi.org/10.1038/s41598-024-82001-w.
21. Dasgupta, S., Zhu, X., Irfan, M. S., Rahman, M., Gong, J., & Jones, S. (2023). AI machine vision for safety and mobility: an autonomous vehicle perspective. *Handbook on Artificial Intelligence and Transport*, 380-409.
22. Bejarbaneh, E. Y., Du, H., & Naghdy, F. (2024). Exploring Shared Perception and Control in Cooperative Vehicle-Intersection Systems: A Review. *IEEE Transactions on Intelligent Transportation Systems*.
23. Srinivasa Gowda G. K, et al., Optimizing the cyber-physical intelligent transportation system network using enhanced models for data routing and task scheduling, Digital Communications and Networks, 2025, https://doi.org/10.1016/j.dcan.2025.01.004.
24. Nabeel S Alsharafa et al., An Edge Assisted Internet of Things Model for Renewable Energy and Cost-Effective Greenhouse Crop Management, Journal of Machine and Computing, Vol. 5, No. 1, pp. 576-588, 2025, https://doi.org/10.53759/7669/jmc202505045.
25. T. Gopalakrishnan et al., Leveraging blockchain technology to combat deception, deepfake, and counterfeit systems, Journal of Discrete Mathematical Sciences and Cryptography, 27:7, 2143–2154, 2024, DOI: 10.47974/JDMSC-2087