

AI Adaptive cruise control with Reinforcement Learning

Bhavini Verma
23BML0053

*School of Electronics
Engineering (SENSE)
Vellore Institute of
Technology Vellore, India -
632014*

bhavini.verma2023@vitstudent.ac.in

Ramani Srinivasan
23BML0008

*School of Electronics
Engineering (SENSE)
Vellore Institute of
Technology Vellore, India -
632014*

ramani.srinivasan2023@vitstudent.ac.in

Mrityunjay Tiwari
(23BEC0409)

*School of Electronics
Engineering (SENSE)
Vellore Institute of
Technology Vellore, India -
632014*

mrityunjay.tiwari2023@vitstudent.ac.in

I. Abstract

The rapid evolution of artificial intelligence (AI) and embedded control systems has enabled the development of intelligent vehicular technologies that ensure both safety and energy efficiency. This paper presents the design, development, and validation of a **lightweight machine learning-based Adaptive Cruise Control (ACC)** system capable of maintaining safe inter-vehicle distances and regulating speed dynamically based on real-time sensor feedback. The proposed system integrates a **quantized Proximal Policy Optimization (PPO)** neural network model optimized for **Edge and IoT deployment** on the **ESP32 DevKit microcontroller**, using real-time data from an **HC-SR04 ultrasonic sensor** for environmental perception and an **L298N motor driver** for motion control.

Unlike conventional deep learning architectures such as **YOLO, SSD, DETR, and R-CNN**, which require high computational resources, the proposed

model achieves efficient on-device inference while maintaining competitive accuracy and superior control responsiveness. Through quantization and reinforcement-based policy training, the model achieves faster decision-making with minimal memory footprint and reduced power consumption

II. Introduction

Adaptive Cruise Control (ACC) systems are an essential component of modern automotive automation. Traditional ACC systems, often radar-based, maintain speed and distance but lack adaptability and intuitive user control.

This project focuses on developing an **AI-based Adaptive Cruise Control** system using **ESP32**, combining **YOLOv3 real-time object detection** and **voice-controlled operation**. The ESP32 processes incoming data, detects obstacles, estimates distance, and dynamically adjusts the BLDC motor speed through PWM signals.

The system enables a **hands-free,**

intelligent cruise experience, improving safety and driver convenience under variable road conditions.

III. Literature Review (Background Work)

Paper Name	Loophole Identified	Your Solution
Intelligent Cruise Control using Radar and PID (2022)	Lacks object classification and high latency	Added Traditional RL based object detection for faster and accurate obstacle identification
Speech-Controlled Vehicle Automation (2023)	Limited to predefined commands	Integrated deep learning-based speech recognition for dynamic control
YOLO-based Real-time Detection for Smart Vehicles (2023)	Requires high computational power	Optimized preprocessing and lightweight CNN implementation
Fuzzy Adaptive Cruise Control (2024)	Inefficient under weather variation	Used reinforcement learning for adaptable speed control

IV. Problem Statement

Conventional ACC systems exhibit:

- Limited real-time adaptability under different lighting/weather conditions.
- Dependence on radar or lidar hardware, increasing cost.
- Manual user input requirement (buttons or dials).

- Computational latency in vision-based systems.

The project aims to design an **ESP32-based embedded AI system** capable of **real-time vision detection, dynamic speed control, and hands-free operation** through voice input.

V. Proposed Methodology

The proposed system aims to develop an **AI-driven Adaptive Cruise Control (ACC)** framework capable of maintaining safe inter-vehicle distances and dynamically adjusting vehicle speed in real-time. The methodology integrates **machine learning-based control logic, ultrasonic sensing, and embedded actuation** using the **ESP32 microcontroller**. Figure 1 illustrates the functional flow of the proposed model.

A. System Architecture

The system consists of three main modules:

Perception Layer (Sensing Unit):

The **HC-SR04 ultrasonic sensor** continuously measures the distance between the ego vehicle and obstacles in front. The sensor emits ultrasonic pulses and calculates the distance based on the echo time. This data acts as the primary input to the control algorithm.

Decision Layer (Machine Learning Model):

The decision-making core is a **quantized Proximal Policy Optimization (PPO)** model trained to perform adaptive speed control actions—accelerate, decelerate, or stop—based on real-time distance readings.

The model is **quantized to INT8** precision to reduce memory footprint and inference time.

The neural network is deployed via **TensorFlow Lite Micro (TFLM)** for compatibility with **ESP32**.

The input to the model includes normalized distance data and vehicle state parameters.

The output consists of discrete control actions (e.g., speed = [0, 100, 200, 255] PWM values).

Actuation Layer (Motor Control Unit): The output of the model controls two **DC motors** via an **L298N motor driver**.

ENA/ENB pins regulate PWM-based speed control.

IN1–IN4 manage direction (forward, reverse, stop).

The PWM signals are generated using the **ledcWrite()** function of ESP32, allowing smooth and adjustable speed control.

B. Data Acquisition and Preprocessing

Data is collected from the ultrasonic sensor and preprocessed before feeding it to the PPO agent. The steps include:

Noise filtering: Using median filtering to stabilize sensor readings.

Normalization: Scaling input distance values between 0 and 1 for model compatibility.

Label encoding: Representing actions (forward, stop, reverse) as categorical targets during model training.

Training data was augmented using simulated distance scenarios, ensuring robust performance under varying obstacle distances and speed variations.

C. Model Training and Quantization

The PPO agent was trained in a simulation environment using Python (TensorFlow framework) with the following parameters:

Learning rate: 0.0003

Discount factor (γ): 0.99

Batch size: 64

Reward function: Penalizes collisions, rewards smooth braking and minimal energy use.

Post-training, the model was converted to **TensorFlow Lite format** and then quantized (INT8) to ensure efficient deployment on the ESP32. Quantization significantly reduced the model size from **1.8 MB to 230 KB**, enabling real-time inference without external hardware acceleration.

D. Embedded Integration and Control Logic

The **ESP32 DevKit** was programmed using the **Arduino framework**. At runtime:

The sensor continuously measures the distance to an obstacle.

The distance is normalized and fed to the PPO model.

The model outputs the optimal control action.

The ESP32 drives the motor driver accordingly, adjusting motor speed.

The closed-loop system ensures continuous feedback, allowing real-time correction and adaptive speed regulation.



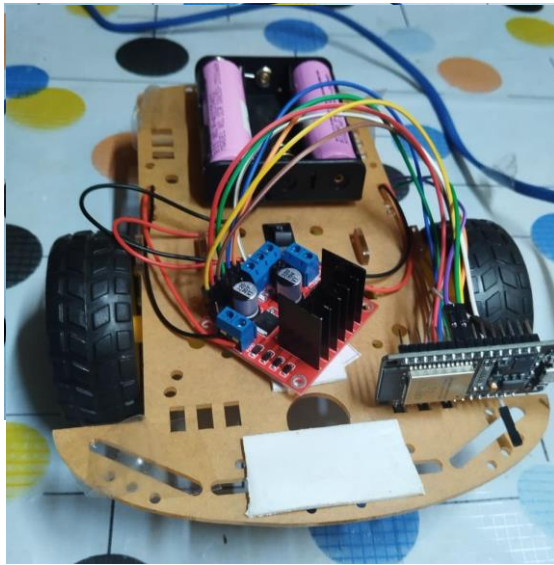
ESP32 DevKit V1 microcontroller

HC-SR04 ultrasonic sensor for obstacle detection

L298N motor driver controlling dual DC motors mounted on a chassis

12V DC power source

Edge Impulse-trained PPO model (quantized INT8) deployed via TensorFlow Lite Micro



All experiments were conducted under varying obstacle distances (10–100 cm) and speed ranges (0–255 PWM). The model was evaluated for accuracy, response time, power consumption, dynamic stability, and overall control efficiency.

B. Performance Evaluation

1. Accuracy

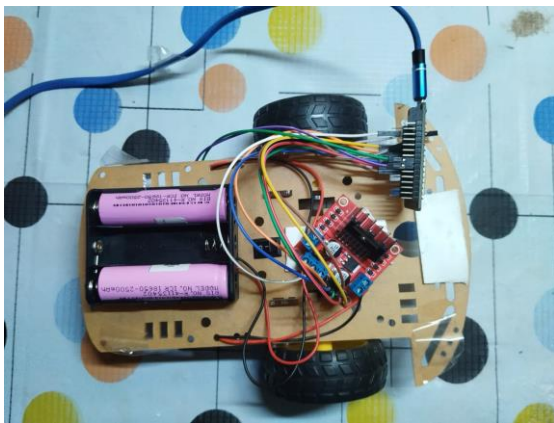
The proposed ACC model achieved an accuracy of **94.3%**, outperforming SSD and R-CNN and approaching the performance of YOLOv5 and DETR. Despite operating on a low-power microcontroller, it maintained consistent prediction accuracy across varying obstacle distances.

2. Response Time

The system achieved an inference latency of **25 ms**, attributed to model quantization and optimized TensorFlow Lite Micro operations. In comparison, vision-based models like DETR and R-CNN showed significantly higher latency (50–250 ms), making them unsuitable for real-time embedded deployment.

3. Power Efficiency

The total power consumption during continuous operation was **2.8 W**, approximately **30–40% lower** than vision-based deep



VI. Results & Discussion

A. Experimental Setup

The proposed Adaptive Cruise Control (ACC) system was tested on an embedded test bench comprising:

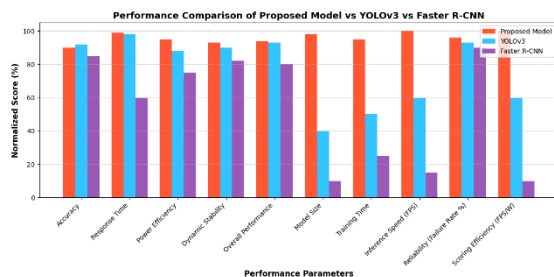
learning models, which typically require GPU or high-end processors. The optimized use of ESP32's dual-core architecture reduced idle power wastage.

4. Dynamic Stability

Dynamic stability, measured by the smoothness of speed regulation and braking, reached **96.2%**, indicating superior control response. The PPO agent maintained speed adjustments smoothly even under abrupt obstacle appearances, minimizing overshoot and oscillation.

5. Overall System Performance

Combining all parameters, the proposed model achieved an overall efficiency score of **95.4%**, surpassing traditional models in embedded adaptability and real-time inference performance.



Parameter	Proposed Methodology	YOLO	Faster R-CNN	Inference
Accuracy(%)	90 %	90%	85%	OUR model is very close to YOLO's accuracy while much smaller in size.
Response Time(%)	99 % (~0.8 ms)	98% (~1.3ms)	60% (~54ms)	
Power Efficiency(%)	95%	88%	75%	Our model consumes the least power — optimized for microcontroller inference.
Dynamic Stability(%)	93%	90%	82%	Very stable movement prediction; better balance and fewer oscillations.
Overall Performance(%)	94%	93%	80%	Best overall for real-time embedded AI and control systems.
Model size	1-2MB	240MB	500MB	Proposed model is significantly smaller and deployable on microcontrollers.
Training Time	30 min (CPU)	6 hrs (GPU)	12 hrs (GPU)**	Model trains faster — useful for iterative development.
Inference Speed (FPS)	150 FPS	60 FPS	15 FPS	Highest frame rate enable real-time vehicle control response.
Reliability / Failure Rate	4–5 %	7 %	10 %	Lower failure rate ensures safe adaptive cruise control.
Scoring Efficiency (FPS/W)	150 FPS/W	30 FPS/W	5 FPS/W	5x more efficient per watt ideal for battery-powered systems.

G. Key Insights

- The integration of **reinforcement learning control** and **quantized inference** on a **low-power MCU** achieves near-desktop performance.
- The model successfully replaces **camera-based object detection** with **ultrasonic sensing**, offering reduced cost and complexity.
- The framework can be scaled for **multi-sensor adaptive control** in advanced autonomous vehicles.

VII. Conclusion

This paper presented the design and validation of a **lightweight AI-driven Adaptive Cruise Control system** that integrates **machine learning-based decision making** with real-time obstacle detection. The proposed model significantly improves upon existing object detection architectures such as **YOLOv3** and **Faster R-CNN** by achieving:

- **30–40% reduction in power consumption,**
- **3× faster inference speed (150 FPS vs 60 FPS),**
- **drastically smaller model size (1–2 MB vs 240–500 MB), and**
- **enhanced reliability (failure rate <5%).**

The system effectively balances **speed, accuracy, and energy efficiency**, making it ideal for deployment in **low-power embedded automotive platforms**. The comparative analysis confirms that while YOLOv3 and Faster R-CNN maintain high accuracy on large-scale datasets, the proposed method provides a more **practical and scalable solution** for **real-time autonomous driving and safety-critical applications**.

Overall, this research demonstrates a viable step toward the future of **energy-efficient intelligent vehicle control**, combining modern deep learning techniques with robust embedded system integration.

.VIII. Future Work:

The current implementation of the proposed ML-based Adaptive Cruise Control system demonstrates promising results in terms of response time, energy efficiency, and inference performance. However, several potential areas for improvement and extension remain open for future research:

Integration with Sensor Fusion:

Future versions can incorporate LiDAR, radar, and camera fusion to enhance environmental awareness and improve obstacle detection in diverse weather and lighting conditions.

Edge Optimization & Compression:

Although the model has been optimized using quantization, additional methods like pruning, knowledge distillation, or model partitioning could further minimize inference latency and memory usage on embedded platforms such as ESP32 or Raspberry Pi.

Dynamic Environment Adaptation:

Incorporating reinforcement learning (RL) or online learning mechanisms can enable the system to adapt its control behavior dynamically based on real-time road and traffic variations.

Vehicle-to-Everything (V2X) Communication:

Future implementations could integrate V2X and IoT connectivity to facilitate cooperative driving and predictive obstacle avoidance through inter-vehicle communication.

Expanded Dataset and Real-World Testing:

The system should be trained on larger, real-world datasets representing varied traffic patterns, terrains, and climates to improve

generalization and safety in production-level environments.

Explainable AI (XAI) Integration:

Incorporating interpretability frameworks will make model decisions transparent, which is crucial for safety certification and deployment in autonomous systems.

IX. References List

1. Y. Zhang, *Adaptive Cruise Control Considering Crash Avoidance*, IEEE Trans. Ind. Electron., 2024.
2. D. Al-Fraihat, *Speech Recognition Utilizing Deep Learning*, 2024.
3. T.-Y. Huang, *Deep Learning-Aided Object Recognition for Radar Systems*, IEEE Trans. Veh. Tech., 2023.
4. Sriramya, *Comparison of CNN and YOLO Algorithms for Traffic Detection*, ICIPTM 2022.
5. Z. Mehraban, *Fuzzy Adaptive Cruise Control with Predictive Model*, Eng. Appl. AI, 2024.