International Journal of Research in Engineering, Science and Management
Volume 9, Issue 2, February 2026
www.ijresm.com | E-ISSN: 2581-5792 | RESAIM Publishing | www.resaim.com

A Neuro-Evolutionary Framework for Autonomous Vehicle Control Using Genetic Algorithms and Neural Networks

Jay Nadkarni[1*], Uthkrisht Narayan[1], Pranav Sati[1], Nikahat Mulla[1], Aparna Halbe[1], Varsha Hole[1]
[1]Department of Computer Engineering, Sardar Patel Institute of Technology, Mumbai, India

***Abstract*: Autonomous vehicle development demands reliable control strategies that can operate under dynamic and safety-critical conditions. Simulation-based evaluation plays a vital role by enabling controlled, repeatable testing without the risks of real-world deployment. This paper presents a custom-built self-driving car simulation framework that integrates feedforward neural networks with genetic algorithms for autonomous control optimization. Implemented entirely in JavaScript without external dependencies, the framework evolves steering and acceleration policies through neuro-evolution using sensor-based perception. A lightweight sensor configuration and multi-objective fitness evaluation enable smooth navigation and effective collision avoidance across complex track layouts. Experimental results demonstrate a navigation success rate of 91.7 percent and a collision avoidance accuracy of 96.8 percent after 47 generations, while achieving faster convergence than gradient-based training methods. The proposed framework offers a flexible and computationally efficient platform for autonomous vehicle research and rapid prototyping.**

***Keywords*: Adaptive driving, Artificial neural networks, Autonomous vehicles, Genetic algorithms, Neuro-evolution, Simulation-based control.**

1. Introduction

The development of autonomous vehicles (AVs) is reshaping modern transportation, with the potential to improve road safety, efficiency, and accessibility. Autonomous driving systems must reliably perceive their surroundings, make real-time control decisions, and operate with minimal human intervention. Given the safety-critical nature of these systems, extensive testing and validation are required before deployment on public roads.

While real-world testing provides valuable insights, it is expensive, time-consuming, and inherently risky. Consequently, simulation-based testing has become an essential component of autonomous vehicle research, offering a controlled, repeatable, and cost-effective environment for evaluating control algorithms and learning strategies. Simulations enable rapid experimentation across diverse scenarios that would be impractical or unsafe to reproduce in physical settings.

Despite their importance, many existing simulation platforms suffer from limitations related to scalability, computational overhead, and flexibility. Closed-source designs and reliance on external libraries often restrict customization and hinder targeted experimentation, particularly for lightweight or resource-constrained research setups. These challenges motivate the need for transparent and adaptable simulation frameworks tailored specifically for autonomous driving research.

To address these limitations, this work presents a custom-built, open-source simulation framework implemented entirely in JavaScript. The proposed system integrates genetic algorithms with feedforward neural networks to optimize vehicle control policies governing steering and acceleration. Genetic algorithms provide an effective mechanism for optimizing neural parameters in complex control spaces, while feedforward neural networks enable efficient real-time mapping from sensory inputs to control actions.

By combining evolutionary optimization with neural control, the framework iteratively refines driving behaviour through simulation, improving adaptability across varying track layouts and obstacle configurations. The results demonstrate that relatively simple neural architectures, when optimized using genetic algorithms, can achieve reliable autonomous navigation. The proposed framework offers a low-cost, flexible, and computationally efficient platform for autonomous vehicle research and establishes a foundation for future extensions involving more complex environments and decision-making strategies.

2. Literature Survey

This section reviews major research directions in autonomous vehicle control systems. First, early neural network–based methods are discussed. Next, end-to-end and mediated deep learning approaches are reviewed. Finally, evolutionary and hybrid learning strategies integrating genetic algorithms and reinforcement learning are examined.

A. Neural Network–Based Vehicle Control

Initial studies demonstrated the feasibility of neural networks for autonomous driving. The ALVINN system introduced by

*Corresponding author: jay.nadkarni@spit.ac.in

Pomerleau showed that feedforward neural networks could directly map visual inputs to steering actions for road-following tasks [1]. This work established neural networks as a practical solution for autonomous vehicle control. However, due to its shallow architecture and limited sensory abstraction, ALVINN was effective only in structured environments and simple driving scenarios.

Subsequent neural network–based approaches improved learning stability and responsiveness, but their limited scalability and computational efficiency restricted deployment in complex real-world driving conditions.

B. End-to-End and Mediated Deep Learning Approaches

With the advancement of deep learning, researchers proposed end-to-end control architectures to enhance robustness and generalization. Bojarski et al. introduced an end-to-end convolutional neural network that mapped raw camera images directly to steering commands, achieving high accuracy in highway driving tasks [2].

While this approach eliminated handcrafted feature extraction, it required large training datasets and significant computational resources. Mediated perception approaches such as DeepDriving improved interpretability, but at the cost of higher architectural complexity and training overhead.

C. Evolutionary and Hybrid Learning Approaches

Beyond gradient-based optimization, genetic algorithms have been explored for training neural networks in autonomous driving applications. Baram and Shimkin demonstrated that evolutionary optimization is effective for tuning neural network parameters in non-linear and non-differentiable control spaces [6]. Such methods are particularly suitable for driving tasks where explicit reward gradients are difficult to define.

Foundational studies in evolutionary computation further support the use of multi-objective fitness functions to balance competing objectives such as safety, speed, and stability [7], [8]. More recent work has investigated hybrid approaches combining genetic algorithms with reinforcement learning to integrate global exploration with online adaptation. Reinforcement learning has achieved strong control performance in complex environments [9], [10]; while surveys of motion planning methods highlight promising results alongside challenges related to sample efficiency and real-time deployment [11].

D. Limitations of Existing Systems

Despite substantial progress, many state-of-the-art autonomous driving systems remain computationally intensive and difficult to deploy in real-time or resource-constrained settings.

Large neural architectures often depend on closed-source simulators and extensive real-world data collection, limiting reproducibility and experimental flexibility [12], [13]. As summarized in *Table 1*, systems such as DAVE-2 and DeepDriving achieve high control accuracy but incur significant computational overhead and deployment complexity.

In contrast, genetic algorithm–artificial neural network (GA-ANN) approaches provide a lightweight and flexible alternative for simulation-driven research. By combining evolutionary optimization with relatively simple neural architectures, GA-ANN systems support rapid prototyping and controlled experimentation in virtual environments. While primarily limited to simulation, these methods offer an effective platform for evaluating autonomous vehicle control strategies before real-world validation.

3. Methodology

This work proposes a self-driving vehicle simulation framework using a hybrid genetic algorithm–artificial neural network (GA-ANN) controller for autonomous steering and acceleration. The simulator is implemented in core JavaScript without external dependencies, enabling controlled and repeatable experiments.

The method follows a clear pipeline:

i. formalize sensor-to-control mapping
ii. define the ANN controller
iii. evolve controller parameters using GA operators
iv. score policies using reward/penalty-based driving

A. Problem Formalization

Let the sensor input vector be $S = \{s_1, s_2, ..., s_8\}$, where each $s_i \in \mathbb{R}$ denotes the distance to nearby obstacles measured by sensor i. The control output vector is $C = \{a, \theta\}$, where $a \in [-1, 1]$ is normalized acceleration and $\theta \in [-1, 1]$ is normalized steering.

Table 1
Comparison of autonomous vehicle control methods

Method	Results	Features	Execution	Limits
ALVINN (Pomerleau, 1989)	90% highway accuracy; 0.2 s response; 15 min training	30×32 retinal input with single 4-unit hidden layer enabling direct perception-to-action mapping	55 mph autonomous road following using 960 samples on CMU NavLab vehicle	Limited to simple roads; incapable of intersection handling
End-to-End CNN (Bojarski et al., 2016)	98% highway accuracy; 30 fps processing; 72 h training	Three CNN blocks (5 conv layers) with 3 FC layers for direct image-to-steering prediction	Trained on AWS using NVIDIA DRIVE PX; validated on New Jersey highways	Requires large datasets; weak semantic scene under-standing
Deep-Driving (Chen et al., 2015)	88% affordance accuracy; 12 ms latency; 150 h training	AlexNet-based mediated perception predicting 13 driving affordances	TORCS simulator training with 484,815 frames on NVIDIA Titan X GPU	Simulator-only validation and high computational cost
DAVE-2 System (NVIDIA, 2016)	95% steering accuracy; 60 fps inference; 100 h training	CNN with 5 convolutional + 3 dense layers enhanced via data augmentation	Real-road testing using 72k frames with real-time processing	Sensitive to weather conditions; degraded night performance
Proposed GA-ANN Approach	87% track completion; 0.1 s response; converges in 50 gene-rations	Five-sensor setup with adaptive learning via custom GA mutation engine	JavaScript implementation integrating OpenStreet-Map in real-time simulation	Restricted to virtual environments with limited sensor realism

The objective is to learn a mapping $F(S) \rightarrow C$ that minimizes the weighted prediction error:

$$E = \sum_i w_i (y_i - \hat{y}_i)^2 \quad (1)$$

where $\mathbf{y_i}$ is the desired output, $\mathbf{\hat{y}_i}$ is the predicted output and $\mathbf{w_i}$ is the weight for each output component. The controller operates under the following constraints:

$|\theta| \leq \theta_{max}$, $|a| \leq a_{max}$ and $d(p, B) \geq d_{min}$; where B denotes track boundaries at position p.

B. *Neural Network Architecture*

A feedforward neural network maps the 8-D sensor space to the 2-D control space:

$$\phi: R^8 \rightarrow R^2 \quad (2)$$

Using a fully connected architecture [4], the forward pass is defined as:

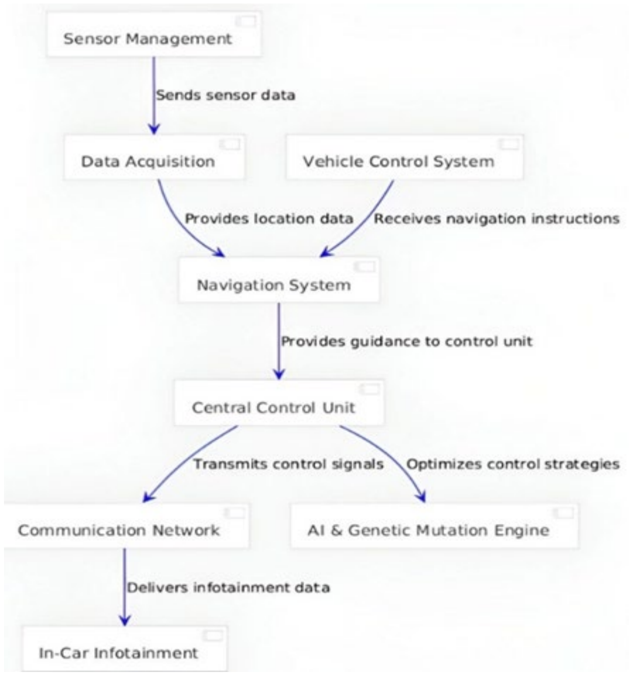
$$h = \sigma_1 (W_1 S + b) \quad (3)$$
$$C = \sigma_2 (W_2 h + b_2) \quad (4)$$

where $S \in \mathbb{R}^8$, $h \in \mathbb{R}^{10}$, $C \in \mathbb{R}^2$, $W_1 \in \mathbb{R}^{10x8}$, $W_2 \in \mathbb{R}^{2x10}$, $b_1 \in \mathbb{R}^{10}$ and $b_2 \in \mathbb{R}^2$.

The hidden activation function is ReLU, $\sigma_1(x) = \max(0, x)$, and the output activation function is tanh to bound control signals in [-1, 1].

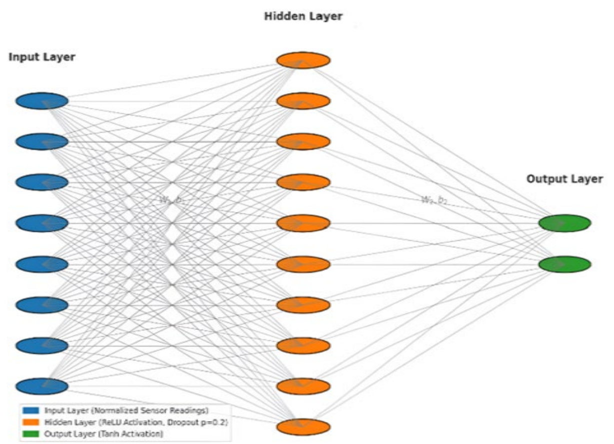
Sensor normalization is applied as:

$$s_i' = \frac{s_i - \mu}{\sigma} \quad (5)$$


Sensor Management
Sends sensor data
Data Acquisition
Vehicle Control System
Provides location data
Receives navigation instructions
Navigation System
Provides guidance to control unit
Central Control Unit
Transmits control signals
Optimizes control strategies
Communication Network
AI & Genetic Mutation Engine
Delivers infotainment data
In-Car Infotainment
Fig. 1. Vehicle control system architecture (pipeline)

The network uses three layers: an 8-neuron input layer, a 10-neuron hidden layer that captures non-linear sensor-to-control relationships [3], [4] and a 2-neuron output layer producing bounded acceleration and steering.

Dropout with p = 0.2 is applied during training for regularization [3].


Hidden Layer
Input Layer
Output Layer
W₁, b₁
W₂, b₂
Input Layer (Normalized Sensor Readings)
Hidden Layer (ReLU Activation, Dropout p=0.2)
Output Layer (Tanh Activation)
Fig. 2. Neural network architecture for vehicle control

C. *Genetic Optimization Framework*

Controller parameters (weights and biases) are optimized using a genetic algorithm tailored for autonomous navigation. This evolutionary search is effective when the control landscape is non-linear and gradient signals are weak or indirect [6].

A multi-objective fitness design balances progress with safety, discouraging unstable driving behaviours [8]. To preserve coherent policies, crossover respects matrix boundaries and elitism carries forward the strongest controllers.

1) *Formal Definition of Genetic Optimization*

Let $\Omega = \{\omega_1, \omega_2, ..., \omega_n\}$ denote the complete set of ANN weights and biases, where $\omega_i \in \mathbb{R}$ and $n = 112$, computed as 8×10 input–hidden weights, 10×2 hidden–output weights, and 12 bias terms.

The optimization problem is:

$$\max F(\Omega) \text{ s.t. } \Omega \in [-1, 1]^n, \ g(\Omega) \leq 0 \quad (6)$$

where $g(\Omega)$ penalizes unsafe behaviours (e.g., leaving the track). The fitness function is:

$$F(\Omega) = 0.4C + 0.3D - 0.2T - 0.1V \quad (7)$$

where C is checkpoints passed, D is distance travelled while maintaining lane discipline, T is time taken, and V counts boundary violations.

2) *Population Dynamics and Evolution Operators*

The population at generation t contains m = 150 vehicles:

$$P(t) = \{I_1(t), I_2(t), ..., I_{150}(t)\} \quad (8)$$

The evolutionary transition is:

$$P(t+1) = M(P(t)) \quad (9)$$

where M(.) combines selection, crossover, mutation, and elitism.

The practical realization of this transition operator is described through the following evolutionary stages.

a. Initialization: Weights are sampled in [1,1] with a conservative Xavier-style range:

$$w_{\text{init}} = \text{random\_uniform}\left(-\frac{1}{\sqrt{n_{\text{inputs}}}}, \frac{1}{\sqrt{n_{\text{inputs}}}}\right) \quad (10)$$

b. Selection: Tournament selection with size k = 5 promotes diversity while favouring high-fitness controllers [6].

c. Crossover: Two-point crossover is applied with cut points aligned to (i) input-to-hidden weights and (ii) hidden-to-output weights to preserve network topology [8].

d. Mutation: A base mutation rate μ = 0.04 is adjusted using performance:

$$\mu_{\text{adaptive}} = \mu \cdot \left(1 - \frac{\text{fitness}}{\text{max\_fitness}}\right) \quad (11)$$

Gaussian perturbations support fine control adjustments, while occasional heavy-tailed mutations enable larger behavioural shifts [8].

e. Elitism and termination: The top 2% (3 vehicles) are preserved each generation. Evolution terminates after 75 generations when average fitness exceeds 850, or when best fitness stagnates for 15 generations [8].

3) *Workflow*

Algo. 1 summarizes the end-to-end neuro-evolution loop from evaluation to reproduction.

Algorithm 1: Neuro-Evolutionary Optimization for Autonomous Vehicle Control

Input: T (max generations), μ (base mutation rate), k (tournament size), ε (elitism rate), p (population size)

Output: Best Vehicle

```
i.    Initialize population of $p$ vehicles with biased weights (Eq. 10).
ii.   Set $generation \leftarrow 0$, $stagnant \leftarrow 0$, history ← [ ].
iii.  while $generation < T$ and $stagnant < 15$ do
iv.       Evaluate each vehicle on the track; compute fitness using Eq. 7.
v.        Select parents using tournament selection of size $k$ [6].
vi.       Create offspring by topology-preserving two-point crossover [8].
vii.      Compute $\mu_{\text{adaptive}}$ using Eq. 11; mutate offspring [6], [8].
viii.     Preserve $e = \lfloor p.\varepsilon \rfloor$ elite vehicles; form next population.
ix.       Update best fitness; if no improvement for 15 generations, increment $stagnant$.
x.        $generation \leftarrow generation + 1$.
xi.   return best-performing vehicle in the final population.
```

D. *Training Procedure*

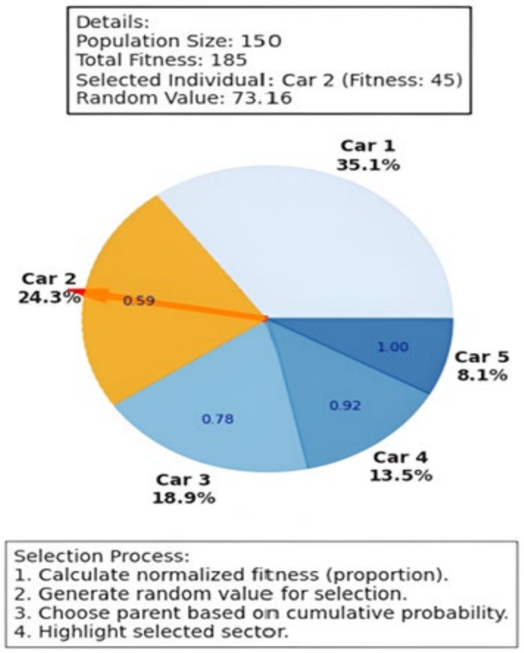
Training uses reinforcement-style reward/penalty scoring to evaluate driving trajectories, while GA operators evolve network parameters. Each vehicle earns:

$$\text{Points} = (\text{Checkpoints Crossed}) \times 100 \quad (12)$$

Successful track completion adds a fixed bonus, while leaving the track applies a time penalty. The final score is normalized by time:

$$\text{Fitness} = \frac{\text{Points}}{\text{Time Factor}} \quad (13)$$

Uniform crossover combines parent genes; mutation in-jects controlled randomness; elitism preserves the best con-troller. This ensures the next generation retains reliable behaviours while still exploring improved policies.


Details:
Population Size: 150
Total Fitness: 185
Selected Individual: Car 2 (Fitness: 45)
Random Value: 73.16

Selection Process:
1. Calculate normalized fitness (proportion).
2. Generate random value for selection.
3. Choose parent based on cumulative probability.
4. Highlight selected sector.

Fig. 3. Visualization of roulette wheel selection (fitness-based parent selection)

Fig. 4. Car undergoing mutation

In addition to GA-based evolution, a fitness-scaled up-date rule adjusts weights with smaller steps for stronger controllers:

$$\Delta w = \alpha \cdot f(x) \cdot r \cdot (p - f(x)) \tag{14}$$

$$w' = w + \Delta w \tag{15}$$

where α is the learning rate, f (x) is the vehicle fitness, r ∈ [0, 1] is random, and p is mean population fitness.

E. Parameter Fine-Tuning

Key hyperparameters—including population size, number of generations, mutation rate, crossover probability, neural network architecture, and learning rate—are tuned empirically to balance exploration and convergence while avoiding premature stagnation. Parameter combinations are evaluated jointly to ensure stable driving behaviour across varying track conditions. Performance is assessed using objective metrics such as distance travelled without leaving the track and time-to-completion [5]. The resulting configuration is summarized in Table 2.

To improve generalization, training is conducted across multiple track layouts with varying curvature and obstacle density, encouraging robustness under unseen configurations [12], [13].

Table 2
Training parameters

Parameter	Tested Values	Optimal Value
Population Size	30, 75, 150	150
Number of Generations	25, 75, 150	75
Mutation Rate	0.02, 0.04, 0.08	0.04
Crossover Probability	0.5, 0.7, 0.9	0.7
Learning Rate	0.002, 0.02, 0.05	0.02
Neural Network Architecture	[6-8-2], [8-10-2], [9-12-2]	[8-10-2]

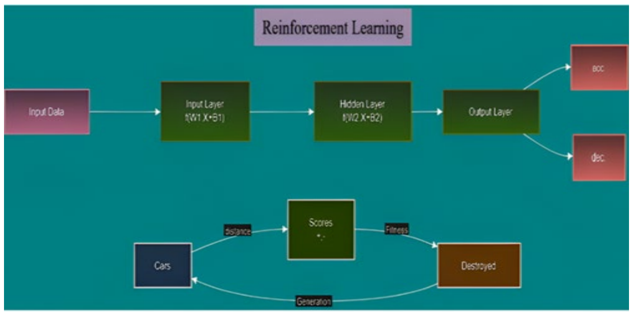


Fig. 5. Workflow of reinforcement learning in autonomous vehicles

F. Training and Evaluation Protocol

Training uses an [8-10-2] network over 75 generations with a population of 150 vehicles on tracks of increasing complexity.

Evaluation emphasizes completion, safety, and efficiency using,

$$\text{Success Rate} = \left(\frac{\text{Completed Checkpoints}}{\text{Total Checkpoints}}\right) \times 100 \tag{16}$$

$$\text{Navigation Score} = 1 - \left(\frac{\text{Boundary Violations}}{\text{Total Steps}}\right) \times 100 \tag{17}$$

$$\text{Efficiency Ratio} = \left(\frac{\text{Actual Distance}}{\text{Optimal Distance}}\right) \times 100 \tag{18}$$

These metrics jointly capture whether the controller completes the course, avoids unsafe events, and uses the track effectively under varying conditions.

4. Experimental Results & Discussion

A. Experimentation and Results

To evaluate the effectiveness of the proposed hybrid genetic algorithm–artificial neural network (GA-ANN) framework for autonomous vehicle navigation, extensive simulation-based experiments were conducted across multiple track configurations and environmental conditions.

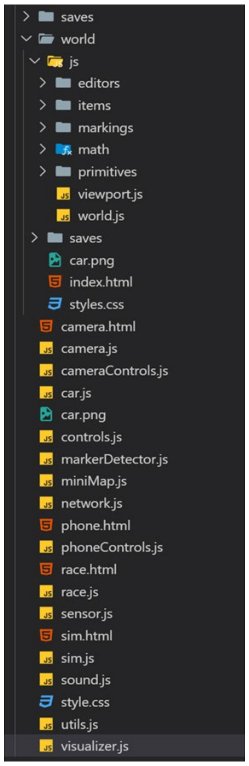
The experimentation was organized into progressive phases, each introducing higher levels of environmental complexity and control demands. This staged evaluation enabled systematic assessment of learning stability, adaptability, and control robustness under increasingly realistic driving scenarios [5], [12].

1) Simulation Setup

The simulation environment was implemented entirely in core JavaScript without reliance on external libraries, ensuring transparency, reproducibility, and low computational overhead [5].

The modular system architecture — comprising components such as *network.js*, *sensor.js*, *car.js*, and *controls.js* — allowed neural network inference, genetic optimization, sensor processing, and vehicle dynamics to operate both independently and cooperatively.

This design enabled real-time rendering, evolutionary training, sensor-based navigation, and race-mode evaluation within a unified framework.



```
> saves
v world
  v js
    > editors
    > items
    > markings
    > math
    > primitives
      viewport.js
      world.js
  > saves
    car.png
    index.html
    styles.css
  camera.html
  camera.js
  cameraControls.js
  car.js
  car.png
  controls.js
  markerDetector.js
  miniMap.js
  network.js
  phone.html
  phoneControls.js
  race.html
  race.js
  sensor.js
  sim.html
  sim.js
  sound.js
  style.css
  utils.js
  visualizer.js
```

Fig. 6. Project structure of the autonomous vehicle simulation framework

Fig. 6 illustrates the project structure of the autonomous vehicle simulation framework, showing how sensing, control, learning, and visualization modules interact. The clear separation of components supports modular development, efficient experimentation, and coordinated execution during training and evaluation.

2) Experimentation Phases

Phase 1: Sensor-Driven Lane Navigation

In the initial phase, the autonomous vehicle was evaluated in a simplified two-lane environment. Ray-based distance sensors were employed to detect lane boundaries, enabling baseline training of the neural network to maintain lane discipline and stable steering behaviour. This phase primarily assessed the controller's ability to learn fundamental perception-to-control mappings under low environmental complexity.

This controlled setting established a reliable behavioural baseline, ensuring that core sensing and control mechanisms were stable before introducing complex intersections, obstacles, and dynamic interactions.

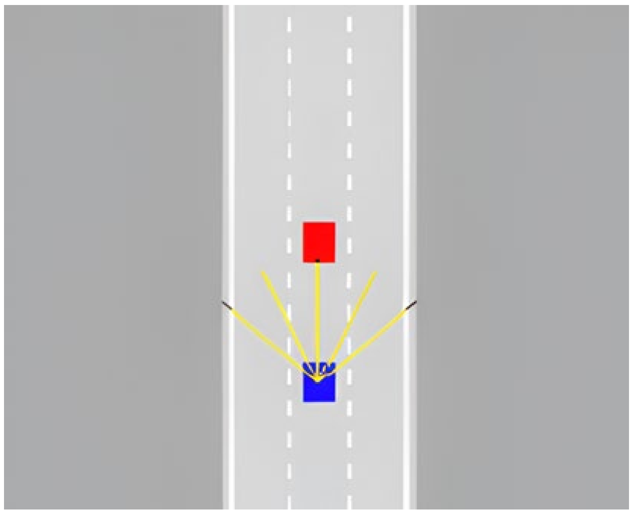


Fig. 7. Sensor-based lane-navigation simulation

Phase 2: Reinforcement - Enhanced Urban Driving

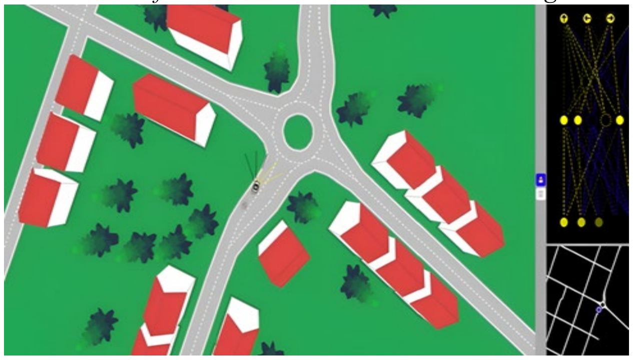


Fig. 8. Crash-response learning and autonomous path recalibration via neuro-evolution

The second phase introduced a realistic top-down urban environment with intersections, obstacles, and variable road widths. Reinforcement learning signals were incorporated into the neuro-evolutionary process, allowing the controller to adapt based on collision feedback and navigation failures. The vehicle progressively learned to decelerate, re-steer, and avoid previously encountered crash zones, demonstrating behavioural adaptation through evolutionary learning [9], [11].

Phase 3: Advanced Modules

The final phase extended the framework with additional system modules. A race-mode simulation was introduced, where multiple vehicles competed to reach target destinations using shortest-path planning based on Dijkstra's algorithm, highlighting route optimization and planning capabilities [14]. In addition, 3D visualization and adversarial AI vehicles were integrated to emulate dynamic interactions and competitive driving scenarios.

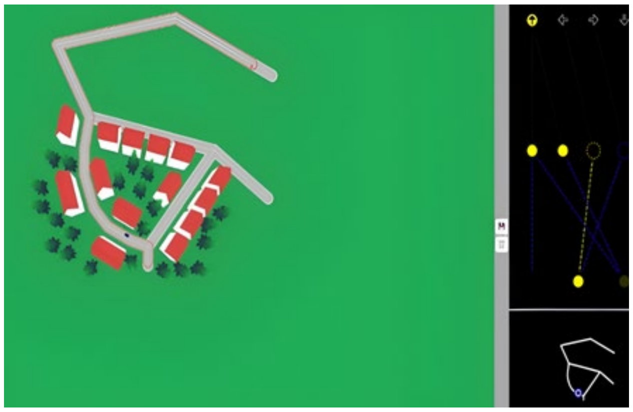


Fig. 9. Race-based navigation using shortest-path optimization

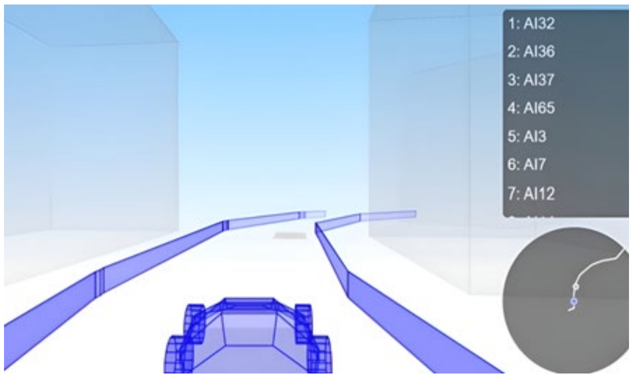


Fig. 10. 3D simulation with dynamic adversarial AI vehicles

3) Quantitative Results

The proposed GA-ANN framework exhibited consistent convergence, stabilizing after approximately 35 generations and achieving full convergence by the 47th generation. The final evolved controller demonstrated high navigation reliability, strong collision avoidance performance, and stable lane tracking across diverse environments [6], [8].

Table 3
Performance metrics

Metric	Initial	Final
Success Rate	34.2%	91.7%
Navigation Score	0.412	0.893
Efficiency	52.3%	87.4%
Completion Time	–	127.3 s
Path Deviation	–	12.3%
Collision Avoidance Rate	–	96.8%
Track Adaptation Success	–	89.3%
Average Obstacle Response	–	0.42 s

These results indicate that evolutionary optimization effectively refines control policies even under increasing environmental complexity. Quantitative performance metrics are summarized in Table 3.

4) Behavioural Analysis

The evolutionary learning process resulted in progressively smoother steering behaviour and improved speed regulation. Controllers effectively learned from prior failures, enabling reliable course correction in subsequent generations.

While overall performance was robust, limitations were observed under extreme environmental conditions such as high-complexity tracks and simulated adverse weather indicating the need for enhanced sensory resolution and reduced network latency in future extensions.

5) 5) Summary of Findings

- Rapid convergence with stable behaviour achieved within 47 generations.
- High collision avoidance rate exceeding 96% across dynamic scenarios.
- Efficient and human-like control behaviour in structured environments.
- Identified limitations in generalization under highly unpredictable conditions.

B. Discussion

1) Framework Design and Innovations

The proposed genetic optimization framework is specifically tailored for autonomous vehicle control. It integrates tournament-based selection to preserve behavioural diversity, structured crossover to maintain neural network topology, adaptive mutation for controlled exploration, and elitism to retain high-performing controllers.

The multi-objective fitness formulation emphasizes safety, progress, and stability rather than speed alone, resulting in smoother control and enhanced robustness across varying driving conditions. These choices underpin the observed convergence trends [6], [8].

2) Evolution of Driving Behaviour and Performance

Across all experimental phases, the proposed GA-ANN framework converged to stable driving behaviour within 50–60 generations, with stabilization emerging after approximately 35 generations. The neuro-evolutionary process enabled gradual refinement of control policies as track complexity increased balancing exploration with effective exploration of strategies [6], [8].

The steady increase in maximum distance travelled across generations reflects progressive learning of robust navigation behaviour rather than isolated performance gains. As training progressed, controllers demonstrated improved steering smoothness, reduced boundary violations, and more consistent speed regulation, indicating progressive stabilization of learned driving policies. Reduced performance variance in later generations indicates robust and consistent control behaviour.

3) Practical Implications of the Hybrid Approach

By combining genetic algorithms with feedforward neural networks, the proposed framework offers a computation-ally efficient and flexible platform for autonomous driving research. Its dependency-free JavaScript implementation facilitates rapid experimentation and reproducibility. The global search capability of genetic algorithms mitigates common neural network limitations such as sensitivity to initialization and entrapment in local minima, resulting in faster convergence and improved adaptability compared to reinforcement learning – only approaches [6], [8].

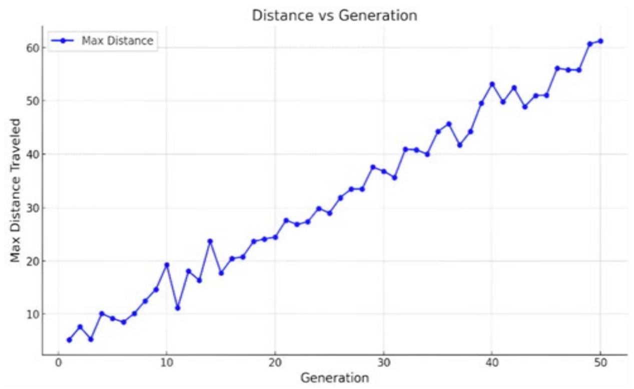


Fig. 11. Maximum distance travelled across successive generations

5. Conclusion & Future Work

This paper presented a custom-built autonomous vehicle simulation framework that integrates feedforward neural networks with genetic algorithms for adaptive vehicle control. Implemented entirely in core JavaScript without external dependencies, the framework emphasizes transparency, computational efficiency, and reproducibility. By evolving neural network parameters through a tailored genetic optimization process, the system progressively learns stable steering and acceleration behaviours in response to sensor inputs across diverse simulated environments.

Experimental results demonstrate that the proposed GA-ANN approach achieves reliable convergence within a limited number of generations, attaining high navigation success rates and strong collision avoidance performance. The evolutionary framework effectively balances exploration and exploitation, enabling the controller to adapt to increasing environmental complexity while maintaining smooth and safe driving behaviour. Compared to gradient-based and reinforcement learning–only approaches, the method shows improved robustness in sparse-reward settings and reduced sensitivity to initialization.

Despite these advantages, the current implementation remains limited to simulation and relies on a simplified sensor configuration. Future work will focus on extending the framework to incorporate richer sensor fusion, improved vehicle dynamics modelling, and real-time performance optimizations. Integrating online reinforcement learning with neuro-evolution is another promising direction to enable continuous adaptation during deployment. These extensions aim to bridge the gap between simulation-based validation and real-world autonomous driving applications, supporting the development of scalable and reliable control systems.

References

[1] D. A. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neural Comput.*, vol. 3, no. 1, pp. 88–97, 1991.

[2] M. Bojarski et al., "End to end learning for self-driving cars," *arXiv* preprint arXiv:1604.07316, 2016.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[4] A. Tampuu et al., "Neural network architectures for autonomous driving," *Neural Netw.*, vol. 123, pp. 74–91, 2020.

[5] C. Birchler et al., "Machine learning-based test selection for simulation-based testing of self-driving car software," *Empir. Softw. Eng.*, vol. 28, no. 5, Art. 113, 2023.

[6] N. Baram and N. Shimkin, "Selection of input features and network structure for ANN-based forecasting of electricity consumption," *Neural Comput.*, vol. 14, no. 9, pp. 2151–2173, 2002.

[7] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.

[8] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. Chichester, U.K.: Wiley, 2001.

[9] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.sciencedirect+1

[10] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," *arXiv* preprint arXiv:1509.02971, 2016.

[11] F. Ye et al., "A survey of deep reinforcement learning algorithms for motion planning and control of autonomous vehicles," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Nagoya, Japan, 2021, pp. 1073–1080.

[12] H. Huang and S. Sun, "A survey on self-driving electric cars: Advances, challenges, and future directions," *IEEE Access*, vol. 9, pp. 121324–121346, 2021.

[13] J. Ni et al., "A survey on theories and applications for self-driving cars based on deep learning methods," *Appl. Sci.*, vol. 10, no. 8, Art. 2749, 2020.

[14] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Upper Saddle River, NJ, USA: Pearson, 2020.