**CAR RACING ENVIRONMENT USING REINFORCEMENT LEARNING USING DQN AND NEAT**

## A PROJECT REPORT

***Submitted By***

**SHIVANESH B - CH.EN.U4AIE21049**

**SUWIN KUMAR J D - CH.EN.U4AIE21056**

**BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING (ARTIFICIAL INTELLIGENCE)**

## Under the guidance of

**R. PRASANNA KUMAR**

 **Submitted to**

**AMRITA VISHWA VIDYAPEETHAM**

**AMRITA SCHOOL OF COMPUTING**

**CHENNAI – 601103**

**October 2024**

## 

**BONAFIDE CERTIFICATE**

This is Certify that this project report entitled **“DYNAMIC RISK ADAPTATION IN HIGH-FREQUENCY TRADING USING HIERARCHICAL REINFORCEMENT LEARNING”** is the bonafide work of **“SHIVANESH B”(CH.EN.U4AIE21049), “SUWIN KUMAR J D”(CH.EN.U4AIE21056),** who carried out the project work under my supervision.

**SIGNATURE**

**R. PRASANNA KUMAR**

Assistant Professor, Department of CSE-AIE.

Amrita School of Computing

Chennai

**INTERNAL EXAMINER**

**ABSTRACT**

The project aims at DQN and NEAT to constitute an efficient racing agent that navigates a car through challenging race tracks. Training on the agent was performed according to the reinforcement learning principles, which means time-dependent decisions regarding steering acceleration and evading obstacles. It uses two separate Q-networks, prediction, and critic to reduce overestimations of action values with the aim of making it efficient in decision-making and learning. NEAT is used to adapt neural network topology through a change in topology across generations in hope of optimizing the performance of the system on different tracks. Unlike the approach above that is far from relying on static datasets, the car agent learns through trial and error. It improves behavior iteratively through feedback through interaction with the racing environment. Therefore, this project, then with the very effective decision-making in DQN accomplishing its tasks and NEAT adaptively evolving architectures, will develop a versatile and efficient solution to autonomous racing. Results Thus, major improvements are appreciated in the performance of agents at different race scenarios.

***Keywords:*** Deep Q-Network (DQN), NeuroEvolution of Augmenting Topology (NEAT), Reinforcement Learning, Autonomous Car Racing, Real-Time Decision Making.

**PROBLEM STATEMENT**

It is hard to simulate autonomous racing agents because at maximum velocity, they have to make heavy decisions regarding steering, acceleration, and collision avoidance in real time. Most classical machine learning techniques are not able to adapt over time due to suboptimal performances in varying new and changing environments. Learning algorithms such as DQN, although highly powerful for decision-making in real-time, still may suffer from a lack of knowing how to generalize effectively over highly variant racing conditions. Definitely designing neural networks which can go on to achieve excellence on complex tracks without requiring manual adaption is added complexity. What is therefore required here is an approach which can combine the efficient decision-making with adaptive neural network evolution.

## PROBLEM SOLUTION

This paper proposes a combination of hybrid approach, deep Q-networks, and neuroevolution of augmenting topology for training the car racing agent. It is therefore deployed in the DQN framework for the ability of the agent to pick at real time by learning experience-based knowledge gained through trial and error with the environment. NEAT evolves the structure of the neural network over multiple generations to adapt the agent to changing complexities within the race track. Thus, this hybrid approach optimizes decision-making and network adaptability so that improvements can be gained in multiple different scenarios in racing.

## 

## OBJECTIVE

* To Design an autonomous car racing agent that would have to make decisions in a complex race track in real time using Deep Q-Network.
* To apply NeuroEvolution of Augmenting Topology (NEAT) to create multi-generational adaptation and optimization of the architecture of a neural network.
* It is trained by interacting with its environment rather than predefined datasets; hence, its adaptability to the race track condition also increases.
* Balance exploration (trying new strategies) and exploitation (using learned strategies) to achieve better performance in speed, coverage of the track, or evading obstacles.
* This work will prove the effectiveness of the combined DQN and NEAT by performance in multiple race track environments regarding completion of laps, speed optimization, and minimal collisions.

**LITERATURE REVIEW**

Noureldin Ragheb and Mervat M.A. Mahmoud in [1] describe an application of DRL for autonomous vehicle control, comparing two algorithms: DQN and DDPG. In this paper the authors mentioned several shortcomings of the supervised learning method when applied to real complexity and its failure to generalize in new or unfamiliar environments. This work establishes, using the CARLA simulator, that while DDPG outperforms DQN in average reward it cannot decrease collisions by as much as DQN. The paper concluded that the DDPG algorithm was more suitable in continuous action spaces in task-based environments but DQN generalized better in discrete environments. It put forth that DRL can provide the potential to develop safer and more reliable autonomous systems in controlling vehicles by mitigating several operational challenges.

Deep Deterministic Policy Gradient-Based Autonomous Driving for Mobile Robots in Sparse Reward Environments" [2] by Minjae Park, Seok Young Lee, Jin Seok Hong, and Nam Kyu Kwon affords the technology to drive mobile robots more effectively in sparse reward environments. The paper suggests that the sparse reward problem can be solved through the application of DDPG with Hindsight Experience Replay. Using sensor information on distance, a TurtleBot3, via ROS-Gazebo simulation, navigates a room. At runtime, HER technique generates new successful experiences from failures of attempts to enhance learning. Therefore, the results show that DDPG with an HER approach enhances performance in both virtual and real-world environments; that is to say, it allows mobile robots to navigate complex spaces much better using fewer reward engineering efforts.

"Q-Learning Algorithms: A Comprehensive Classification and Applications" [3] gives one of the most comprehensive reviews on Q-learning algorithms that can be categorized into either single-agent or multi-agent approaches. It is developed in considerable depth to show rudimentary Q-learning as well from its advanced variation, Deep Q-Learning, Double Q-Learning, and Modular Q-Learning. And pragmatic applications in robotics, game theory, and network control reveal how Q-learning stands at the corner stone of reinforcement learning strategies. It overcomes another weakness of Q-learning that is a big limitation on how it can be used in complex environments; by breaking the walls of storage and convergence through experience replay and hierarchical Q-learning. Analyzing these developments, the paper presents a fair and rounded view of what Q-learning has become for modern artificial intelligence.

"Q-Learning Algorithms: A Comprehensive Classification and Applications," [4] gives one of the most comprehensive reviews on Q-learning algorithms that can be categorized into either single-agent or multi-agent approaches. It is developed in considerable depth to show rudimentary Q-learning as well from its advanced variation, Deep Q-Learning, Double Q-Learning, and Modular Q-Learning. And pragmatic applications in robotics, game theory, and network control reveal how Q-learning stands at the corner stone of reinforcement learning strategies. It overcomes another weakness of Q-learning that is a big limitation on how it can be used in complex environments; by breaking the walls of storage and convergence through experience replay and hierarchical Q-learning. Analyzing these developments, the paper presents a fair and rounded view of what Q-learning has become for modern artificial intelligence.

In this paper, "A New Open-Source Off-Road Environment for Benchmark Generalization of Autonomous Driving," [5] they discuss the weaknesses in today's autonomous driving benchmarks with a new off-road simulation environment. The reason is that most simulators focus on city roads or race tracks and do not test the generalizing ability of driving models in hard and diverse settings. In response to this, they propose a new open-source framework, which would consist of settings for off-roads including forests, deserts, snowfields, and plateaus. The researchers applied Unreal Engine assets in CARLA. Their benchmark enables them to evaluate driving agents in dynamic conditions with high visual diversity. The results indicate that most end-to-end driving methods have failed to generalize well in unseen off-road environments and, hence, provide valuable insights into challenges that face the training of robust autonomous driving systems.

In "Taming an Autonomous Surface Vehicle for Path Following and Collision Avoidance Using Deep Reinforcement Learning", [6] DRL is used to control underactuated ASVs in path following and collision avoidance (COLAV) techniques. It used PPO as the DRL algorithm and trained an AI agent so that it may be outfitted with rangefinder sensors in an OpenAI gym environment. The problem statement presented two objectives: guiding ASV along a predefined path and avoiding collisions with static obstacles. From this research, they were able to obtain almost perfect episodic success rates in several stochastic scenarios, of which the results indicated high potential of PPO for complex real-time marine control applications. It makes important contributions toward marine autonomy, particularly in the process of path-planning tasks with real-time constraint.

The paper "An Empirical Study of DDPG and PPO-Based Reinforcement Learning Algorithms for Autonomous Driving." [7] The authors study here two state-of-art algorithms namely Deep Deterministic Policy Gradient and Proximal Policy Optimization in this paper. In the paper, they conducted their experiment on both single and multi-agent version of the TORCS simulator. It therefore suggests that with regard to PPO DDPG has greater reward and convergence incentives, especially in continuous action space, and can be applied to more autonomous driving tasks, such as lane-keeping and obstacle avoidance, than PPO. Although PPO trains very stably on the initial training, its severe catastrophe suffers from forgetting in a more complex environment. From the authors, it appears that DDPG is useful in such dynamic, multi-agent scenarios mainly because of its off-policy nature and experience replay.

The paper titled "An Empirical Study of DDPG and PPO-Based Reinforcement Learning Algorithms for Autonomous Driving." [8] The authors study here two state-of-art algorithms namely Deep Deterministic Policy Gradient and Proximal Policy Optimization in this paper. In the paper, they conducted their experiment on both single and multi-agent version of the TORCS simulator. It therefore suggests that with regard to PPO DDPG has greater reward and convergence incentives, especially in continuous action space, and can be applied to more autonomous driving tasks, such as lane-keeping and obstacle avoidance, than PPO. Although PPO trains very stably on the initial training, its severe catastrophe suffers from forgetting in a more complex environment. From the authors, it appears that DDPG is useful in such dynamic, multi-agent scenarios mainly because of its off-policy nature and experience replay.

the paper "Cooperative Adaptive Cruise Control Based on Reinforcement Learning for Heavy-Duty BEVs" [9] introduce the novel application of the TD3 algorithm-CACC system for heavy-duty battery electric vehicles. This system represents a decrease in energy consumption due to air drag along with the safety and comfort of the vehicle in platooning. Thus, the authors examined the problem in the context of the energy saving by the reduced intervehicle distance keeping a safe distance by time-to-collision, and headway-based minimum time. The system got tested for highway driving cycles marked energy savings up to 19.82% without violating comfort. This concludes that the proposed CACC system successfully balances energy efficiency with passenger comfort and can well be suited for real-world applications involving heavy-duty vehicles.

New Hierarchical Framework with a Specific Design for Multi-Lane Autonomous Driving Through the Integration of DRL with Rule-Based Methods to Improve Decision Making in Multi-Lane Driving "A Hierarchical Framework for Multi-Lane Autonomous Driving Based on Reinforcement Learning" [10] by Xiaohui Zhang, Jie Sun, Yunpeng Wang, and Jian Sun. The innovation of instantaneous desired speed (IDS) controls the car-following and lane-changing behaviors. IDS originates from high-level DRL, whereas low-level rule-based models enforce particular policies during the execution of the driving policies. The mode optimizes both individual vehicle performances and also the overall traffic flow. Indeed, scenarios such as ring roads and bottlenecks can be more efficient and safer than purely DRL-based and rule-based models. This research work combines DRL with rule-based approaches in order to improve generalization, interpretability, and system-level optimization in traffic.

**METHODOLOGY**

**DATASET OF THE PROJECT**

In the following research work, fixed datasets are not utilized. The agent related to car racing depends upon interactions that are happening within a simulated environment developed using pygame. The agent does not depend on fixed data; it learns from continuous feedback received by the environment. This type of iterative learning allowed the agent to dynamically enhance its performance by judging the result of various actions such as steering, acceleration, and braking plus the smooth navigation through the race track.

Reinforcement learning principles are used in the learning of an agent because of inputs coming from sensors that are "radar points" which will give information about distance from boundaries and obstacles. The agent then changes its decision through real-time feedback. The environment gives a reward to the actions undertaken by the agent in case these actions were successful, for example, staying within the track or completing laps. In unsuccessful attempts, for example collisions, penalties are awarded.

That environment can thus provide direct adaptability for the car agent, and so the process for learning is much more flexible and responsive than would be otherwise possible with static datasets underlying more traditional approaches to supervised learning.

**ALGORITHM AND DESIGN**

In this methodology adopted here, DQN is used for decision-making and NEAT for the architecture of the neural network development. Hence, the complete algorithm can be said to be as follows:

Initialization:

* Initialize a car racing environment with pygame.
* We begin our DQN agent with random weights in its neural network and an empty memory buffer for experience replay.
* Set up the NEAT algorithm for evolving the topology of the neural network.

**Training Loop:**

* Reset the car’s position and state of the track for each episode.
* For each discrete time step, the DQN agent:
  + Accepts distances from the radar sensors to track boundaries as input.
  + Predicts the best action to perform (turn left, go straight, or turn right) using its neural network.
  + Executes the action and tracks the outcome: new radar data, reward, and the status of the car’s operability.
* The NEAT algorithm evolves the network topology based on fitness scores, which consider distance traveled, time alive, and penalties due to collisions.
* The DQN agent stores the experience (state, action, reward, next state) in a memory

buffer.

Periodically, the agent trains its neural network by sampling experiences from the memory buffer and updating the network weights through backpropagation.

Evolution and Optimization:  
NEAT Mutates and evolves the neural network topology using fitness to select top-performing agents and adapt the topology in terms of adding nodes and links.  
  
The network history extends into multiple generations, which further increases the agent's capability to cross the track. Termination: This continues training until such a time that the level of performance attained is adequate enough, whereby the car can execute laps without any form of collisions.

**MODEL ARCHITECTURE**

A diagram of a diagram

Description automatically generated

**MODEL TRAINING**

Methodology of the model used for training it, includes deployment of both DQN and NEAT algorithms. Agent DQN trains within the framework of reinforcement learning, thus being exposed to interaction with the simulated racing environment by accepting instantaneous sensor data, radar information. Based on these inputs, the agent predicts actions steering left, right, or going straight using its neural network. Each action results in a reward or penalty, which helps the agent learn through trial and error.  
  
Additionally, the NEAT algorithm evolves the structure of the neural network by optimizing both the network’s weights and topology. Over multiple episodes, NEAT adjusts the complexity of the network by adding nodes and connections to improve the car’s performance. Throughout the training process, the agent’s experiences (state, action, reward, next state) are stored in a memory buffer, and the network is updated periodically using experience replay and backpropagation to ensure the model improves continuously.

**MODEL EVALUATION**

The model is evaluated based on its performance in the simulated racing environment, with key metrics including distance traveled, time spent on track, and the number of collisions. The agent’s ability to complete laps without going off the track or crashing into obstacles is used to assess its effectiveness. Additionally, fitness scores generated by the NEAT algorithm are used to measure the agent's overall performance.

Evaluation also includes testing the agent’s ability to generalize across different track configurations. This is achieved by running the trained model on new, unseen tracks to assess its adaptability. Success is measured by the agent’s ability to navigate the tracks while maintaining speed, avoiding collisions, and making real-time decisions effectively.

**VISUALIZATION**The pygame simulation allows the user to observe agent performance both when training and evaluating the agent; it theoretically simulates a real-time graphical interface of a car racing environment-it shows car moves, track boundaries, and other obstacles so the user can watch the agent make his or her decisions in real time.  
**Radar Sensors Visualization:**  
The car features radar sensors that detect distances, tracing edges and perhaps obstacles. These are also drawn as lines emanating from the car in the various directions-from which they are updating constantly through the position of the vehicle along with the objects it perceives in its surroundings. This provides for a distinct visual representation of the agent's understanding of its environment.  
**Vehicle Trail Representation:**  
The machine leaves colored tracks tracing along the track. This aids in tracing over time in areas it may pass through and which it may run into something or deviate off the track. Various colors can be used to highlight exploitation, or making use of knowledge gained during practice, and exploration, trying new approaches.

It graphically displays the importance of reward, fitness scores, and loss values over episodes. Therefore, such visualizations give a good insight into what learning trajectory the agent is following the capability of the car will progress as it gains experience.

**RESULTS:**

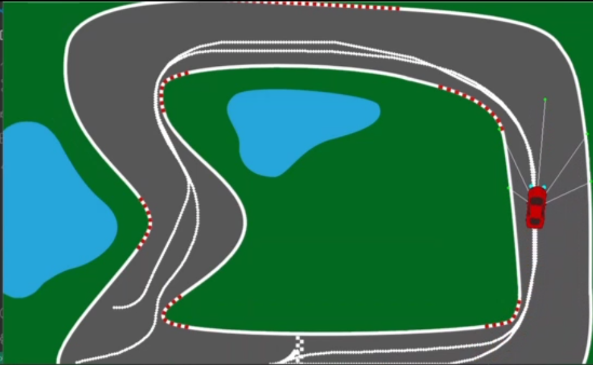
**DQN:**

Figure 1: DQN Agent in Exploration Phase Figure 2: DQN Agent in Exploitation Phase

The figures represent the performance of the vehicle with Deep Q-Network and NeuroEvolution of Augmenting Topology methods. Figure [1] displays the trained DQN agent wherein its reflection on training is reflected in learning by exploring drive patterns that are fitful at times, and multiple deviations are made in paths. This gradually modifies the policy of the agent with time in order to best balance the exploitation and exploration. Figure[2] shows that the exploitation phase of the agent, which has now learned to follow the optimal racing path within minimal deviation to better navigate around the track.

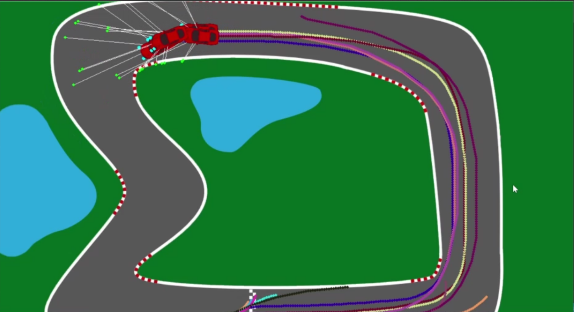


Figure 3: NEAT Evolution Phase Figure 4: NEAT Convergence

Figures [3] and Figures [4] show how the NEAT algorithm drives the evolution of car behavior. Every car is shown with its own trail, which can be compared between neural network architectures at different generations. The multicolored trails reflect the process by which NEAT evolves the network structure in a dynamic fashion in order to settle down to the most optimal one concerning the task. The paths reflect the process through which the neural network evolved from swiftness to control towards a maximum across the track.

**FEAUTRE SCOPE:**

* Multi-Agent Racing: Extend the simulation to include multiple racing agents, allowing for competitive car racing environments where agents learn to navigate not just the track but also other agents.
* Dynamic Track Generation: Implement dynamic and procedurally generated race tracks to test the agent’s adaptability and generalization capabilities on unseen and unpredictable environments.
* Real-Time Performance Improvement. It comes under incorporating real-time feedback mechanisms through which an agent learns to respond in track conditions, either in case of weather or terrain.
* Integration into Physical System: Find out how this simulation is integrated into real autonomous vehicles or racing drones, where the developed model has been applied in real-world environments.

**REFERENCES**

[1] Ragheb, Noureldin & Mahmoud, Mervat. (2024). Implementing Deep Reinforcement Learning in Autonomous Control Systems. Journal of Advanced Research in Applied Sciences and Engineering Technology. 41. 168–178. 10.37934/araset.41.1.168178.

[2] Park, M.; Lee, S.Y.; Hong, J.S.; Kwon, N.K. Deep Deterministic Policy Gradient-Based Autonomous Driving for Mobile Robots in Sparse Reward Environments. *Sensors* **2022**, *22*, 9574. https://doi.org/10.3390/s22249574

[3] B. Jang, M. Kim, G. Harerimana and J. W. Kim, "Q-Learning Algorithms: A Comprehensive Classification and Applications," in *IEEE Access*, vol. 7, pp. 133653-133667, 2019, doi: 10.1109/ACCESS.2019.2941229

[4] B. R. Kiran *et al*., "Deep Reinforcement Learning for Autonomous Driving: A Survey," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909-4926, June 2022, doi: 10.1109/TITS.2021.3054625.

[5] I. Han, D. -H. Park and K. -J. Kim, "A New Open-Source Off-Road Environment for Benchmark Generalization of Autonomous Driving," in *IEEE Access*, vol. 9, pp. 136071-136082, 2021, doi: 10.1109/ACCESS.2021.3116710.

[6] Meyer, Eivind & Robinson, Haakon & Rasheed, Adil & San, Omer. (2019). Taming an autonomous surface vehicle for path following and collision avoidance using deep reinforcement learning. 10.48550/arXiv.1912.08578.

[7] S. Siboo, A. Bhattacharyya, R. Naveen Raj and S. H. Ashwin, "An Empirical Study of DDPG and PPO-Based Reinforcement Learning Algorithms for Autonomous Driving," in *IEEE Access*, vol. 11, pp. 125094-125108, 2023, doi: 10.1109/ACCESS.2023.3330665.

[8] M. U. Yavas, T. Kumbasar and N. K. Ure, "Toward Learning Human-Like, Safe and Comfortable Car-Following Policies With a Novel Deep Reinforcement Learning Approach," in *IEEE Access*, vol. 11, pp. 16843-16854, 2023, doi: 10.1109/ACCESS.2023.3245831

[9] M. Acquarone, F. Miretti, D. Misul and L. Sassara, "Cooperative Adaptive Cruise Control Based on Reinforcement Learning for Heavy-Duty BEVs," in *IEEE Access*, vol. 11, pp. 127145-127156, 2023, doi: 10.1109/ACCESS.2023.3331827.

[10] X. Zhang, J. Sun, Y. Wang and J. Sun, "A Hierarchical Framework for Multi-Lane Autonomous Driving Based on Reinforcement Learning," in *IEEE Open Journal of Intelligent Transportation Systems*, vol. 4, pp. 626-638, 2023, doi: 10.1109/OJITS.2023.3300748.

[11] Xiang, Dantong. (2024). Reinforcement learning in autonomous driving. Applied and Computational Engineering. 48. 17-23. 10.54254/2755-2721/48/20241072.

[12] Ali Irshayyid, Jun Chen, Guojiang Xiong, A review on reinforcement learning-based highway autonomous vehicle control,

Green Energy and Intelligent Transportation,Volume 3, Issue 4,2024,100156,ISSN 2773-1537, https://doi.org/10.1016/j.geits.2024.100156.

[13] Pérez-Gil, Ó., Barea, R., López-Guillén, E. *et al.* Deep reinforcement learning based control for Autonomous Vehicles in CARLA. *Multimed Tools Appl* **81**, 3553–3576 (2022). https://doi.org/10.1007/s11042-021-11437-3

[14] Yanfei Peng, Guozhen Tan, Huaiwei Si, Jianping Li, DRL-GAT-SA: Deep reinforcement learning for autonomous driving planning based on graph attention networks and simplex architecture, Journal of Systems Architecture,Volume 126, 2022, 102505, ISSN 1383-7621, https://doi.org/10.1016/j.sysarc.2022.102505.

[15] Jebessa, Estephanos & Olana, Kidus & Getachew, Kidus & Isteefanos, Stuart & Khan Mohd, Tauheed. (2022). Analysis of Reinforcement Learning in Autonomous Vehicles. 0087-0091. 10.1109/CCWC54503.2022.9720883.