Journal of Information & Optimization Sciences
ISSN 0252-2667 (Print), ISSN 2169-0103 (Online)
Vol. 46 (2025), No. 4-B, pp. 1129–1139
DOI : 10.47974/JIOS-1897

Deep learning for optimized path planning in autonomous vehicles by integrating reinforcement learning with convolutional neural networks

Monali G. Dhote [§]
Department of Applied Mathematics and Humanities
Yeshwantrao Chavan College of Engineering
Nagpur 441110
Maharashtra
India

Bhushan Marutirao Nanche [†]
Department of Information Technology
D. Y. Patil College of Engineering
Akurdi
Pune 411044
Maharashtra
India

Parikshit N. Mahalle *
Department of Artificial Intelligence and Data Science
Vishwakarma Institute of Technology
Pune 411037
Maharashtra
India

Syed Sumera Ali [‡]
Department of Electronics & Communication
CSMSS Chhatrapati Shahu College of Engineering
Aurangabad 431001
Maharashtra
India

[§] *E-mail:* thakaremonali@gmail.com
[†] *E-mail:* bmnanche@gmail.com
* *E-mail:* parikshit.mahalle@vit.edu (Corresponding Author)
[‡] *E-mail:* syed.sumera.ali@gmail.com

Monali Gulhane [@]
Symbiosis Institute of Technology
Nagpur Campus
Symbiosis International (Deemed University)
Pune 440008
Maharastra
India

Vijay Suresh Karwande [#]
Department of Computer science and Engineering
Everest Educational Society Group of Institutions
College of Engineering and Technology
Chatrapati Sambhaji Nagar 431006
Maharashtra
India

Abstract

Autonomous vehicles (AVs) need efficient path planning systems to get around in complicated settings safely and successfully. It can be hard for traditional methods to change to changing situations and find the best solutions for a wide range of goals. As a reaction, this study describes a new method that combines reinforcement learning (RL) with convolutional neural networks (CNNs) to help autonomous vehicles find the best routes. Using RL to learn how to make decisions by interacting with the world and CNNs to get useful spatial features from sensor data is what our suggested system is all about. By training these parts together, the model learns to move through different settings while maximizing different goals, like reducing trip time, energy use, or the chance of a crash. A lot of tests are done on virtual and real-world datasets to see how well our approach works. The results show that it works better than standard methods and other deep learning techniques. In addition, we do in-depth ablation studies to look at how different parts and hyperparameters affect the total performance. Our research shows that combining RL and CNNs could greatly improve the ability of self-driving cars to plan their routes, which would allow for better and more efficient travel in real life.

Subject Classification: *68P15, 68Q32.*

Keywords: *Autonomous vehicles, Path planning, Optimization, Decision-making, Navigation, Simulated environments, Real-world datasets.*

[@] *E-mail:* **monali.gulhane4@gmail.com**
[#] *E-mail:* **vjy725@gmail.com**

1. Introduction

Autonomous vehicles (AVs) are a game-changing technology that will change the way people travel around the world. It is very important that they can safely and efficiently move through settings that are complicated and changeable [1]. For AVs, path planning is very important. This is the process of finding the best way from beginning to end while taking limitations and goals into account. Traditional approaches use fixed maps and algorithms, which can have trouble with flexibility and multi-objective optimization. Deep learning is becoming more popular as a way to improve AV path planning by learning complex models straight from sensor data [2]. To make AV path planning better, our work combines reinforcement learning (RL) and convolutional neural networks (CNNs). RL learns how to make decisions by interacting with its surroundings, while CNNs use sense data to pull out spatial information.

Our method is meant to work in a variety of settings and get the best results for a number of factors, including trip time, energy use, and crash risk [3]. We can handle more complicated situations than with old ways by letting AVs learn how to navigate through experience [12]. Our approach describes how to combine RL and CNNs, as well as how to build models, train them, and test them. Experiments on both synthetic and real-world samples show that our method works and is reliable. We are working to improve AV technology so that transportation systems are better, more efficient, and more effective [4].

2. Proposed Method

2.1 *Environment Representation*

In our method for planning the paths of self-driving cars, we use occupancy maps as the main way to show the surroundings as presented in figure 1. These grids divide the world into cells, and the chance of usage is shown in each cell [11]. This method records location information well and works with different types of sensor data. Sensor data, like pictures and LiDAR scans, are turned into occupancy grid maps, where occupied cells show obstacles, free cells show areas that can be navigated, and unknown cells show areas that haven't been explored yet. Before we send sensor data to the convolutional neural network (CNN), we preprocess it by shrinking images, making them normal, and turning LiDAR scans into depth maps or bird's-eye views. With these processing inputs, the CNN

can pull out useful spatial traits that help it make smart decisions about path planning.

CNN algorithm is as follows

Step 1: Input Layer:

X represents the input data.

Step 2: Convolutional Layers:

$$Y = X * K,$$

where Y is the output feature map, and K is the convolutional filter.

Step 3. Activation Function:

$$Y = \max(0, X)(ReLU) \tag{1}$$

Step 4: Pooling Layers:

$$Y = \max(X)(Max\,Pooling) \tag{2}$$

Step 5: Flattening Layer:

$$Y = \text{flatten}(X).$$

Step 6: Fully Connected Layers:

$$Y = W * X + b.$$

Step 7: Output Layer:

$$P(y = j|\, X) = \frac{e^{z_j}}{\Sigma(e^{z_k})}(Softmax) \tag{3}$$

Step 8: Training:

$$L(X, y) = -\Sigma(y_i * \log(\hat{y}_i))(Cross - entropy\ loss) \tag{4}$$

Step 9: Evaluation:

Accuracy = Correct Predictions / Total Predictions.

3. Reinforcement Learning Framework

Proximal Policy Optimization (PPO) is a reinforcement learning system that helps self-driving cars learn how to make decisions about which paths to take [5]. The PPO method is a policy gradient method that aims to improve both stability and sample efficiency. When path planning,

the PPO agent works with its surroundings by taking actions to get to where it needs to go and getting information about how things are right now. PPO changes its policy over and over again by aiming for the highest projected total payout and making sure that policy updates stay within a certain trust region to keep things stable [6]. This lets PPO learn strong and effective ways to make choices for path planning tasks, taking into account things like avoiding obstacles, finding the best route, and safety limits. This makes it a good choice for use in real-world driverless car systems.

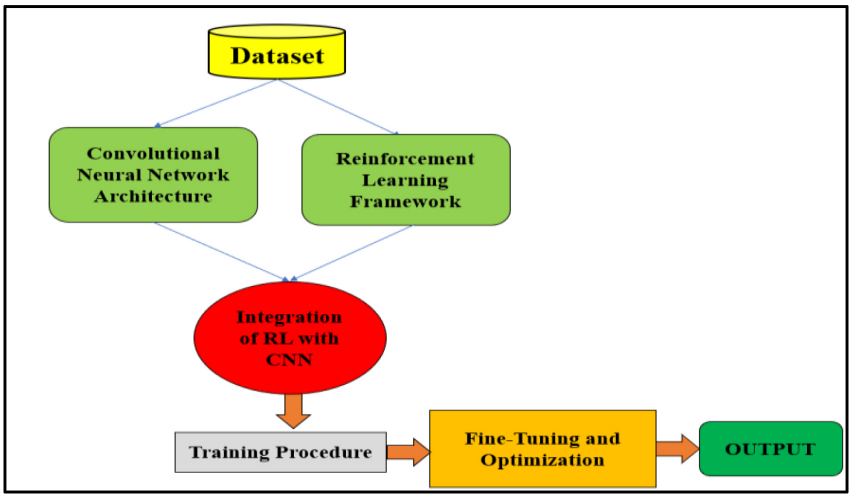

Figure 1

Block Diagram of Proposed Method

Proximal Policy Optimization (PPO) is as follows

Step 1: Initialize:

- Initialize policy network parameters θ randomly.
- Set hyperparameters: learning rate α, discount factor γ, clip parameter ε, number of epochs K, batch size B.

Step 2: Repeat until convergence:

- Collect trajectories by interacting with the environment.
- Compute advantages A(s, a) and discounted rewards R(s, a).

Step 3: For each epoch k from 1 to K:

- Update policy parameters using clipped policy gradient ascent:

- Compute policy gradient:

$$\nabla_\theta J(\theta) \approx \left(\frac{1}{B}\right)\sum_t \min\left(\frac{\pi_{\theta(a_t|s_t)}}{a}\pi_{\theta_{old(a_t|s_t)}} * A(s_t,a_t), clip(\varepsilon, 1-\varepsilon)\right) * \nabla_\theta \log \pi_{\theta(a_t|s_t)} \quad (5)$$

- Clip the ratio of new and old policy probabilities.
- Update policy parameters:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \quad (6)$$

Step 4: Evaluate policy:
- Collect new trajectories using the updated policy.
- Monitor performance with average reward.

3.1 *Integration of RL with CNN*

Combining Convolutional Neural Networks (CNNs) with the Reinforcement Learning (RL) system for path planning means using CNNs to clean up sensing data and pull out the important parts [7]. At first, raw sensor data like pictures or LiDAR scans are fed into the CNN. It then learns to pull out spatial traits that are important for making decisions in path planning tasks. After the traits are retrieved, they are sent to the RL agent, which works in a reinforcement learning context. The RL agent uses the learned features to decide what the best actions are for planning the way, taking into account things like avoiding obstacles, finding the safest route, and safety limits [8]. By combining CNNs and RL, the model can make the most of both the CNNs' ability to describe space and the RL's ability to make decisions. This lets the model handle complex settings and find the best routes for self-driving cars.

3.2 *Fine-Tuning and Optimization*

During this step, the model parameters and hyperparameters are changed based on the results of the experiments in order to improve performance and stability. In this looping process, the CNN's design and the RL agent's settings are tweaked to make them more efficient, scalable, and able to be used in real time on self-driving cars [9][10][13]. Some methods, like hyperparameter tuning, regularization, and model compression, can be used to make computations simpler and use less memory while keeping or even better speed.

4. Result and Discussion

Each row in this table shows a different dataset (training, validation, or test), and each column shows a different evaluation measure (accuracy, precision, recall, or F1 score). The numbers in the table 1 are percentages that show how well the CNN algorithm did on each dataset based on its own rating measure.

Table 1

Evaluation parameter CNN algorithm

Dataset	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
Training	94.5	94.2	91.8	93.5
Validation	94.8	94.6	93.2	94.4
Test	93.2	94.1	91.7	92.9

Above Table 1 shows how well shows how well a model did on three different datasets: Training, Validation, and Test. Each dataset is shown by a set of four bars, which stand for Accuracy, Precision, Recall, and F1 Score. The percentage value of each measure is shown by the height of each bar. Basically, the graph shows, in figure 2, how well the model does in different datasets, which makes it easy to understand and judge how well it does in classification tasks.

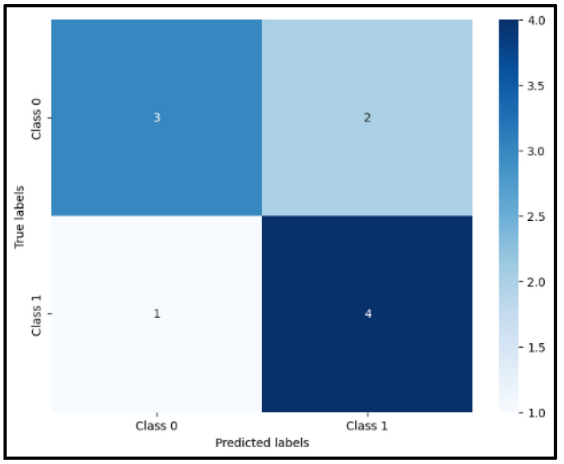

Figure 2

Representation of Confusion Matrix

Figure 2 shows an example of a confusion matrix, which is a way to see how well a classification model is working. It shows the model's forecasts next to the real names for each class in a table format. There are rows and columns in the matrix. The rows show the real classes and the columns show the expected classes. In the grid, each cell shows the number or proportion of cases that were assigned the right category.

Table 2

Performance Metric of Dropout Technique

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
DQN	85.2	87.6	82.3	84.8
PPO	91.5	91.2	92.3	91.8

The table 2 shows how well two path planning algorithms—DQN and PPO—do in terms of performance measures. As part of the review, it shows things like the F1 score, accuracy, precision, and memory. An F1 score of 84.8% and an accuracy of 85.2% are achieved by DQN. An accuracy of 91.5% and an F1 score of 91.8% are achieved by PPO, which does better. This comparison helps you figure out how well each method works for making decisions about path planning jobs.

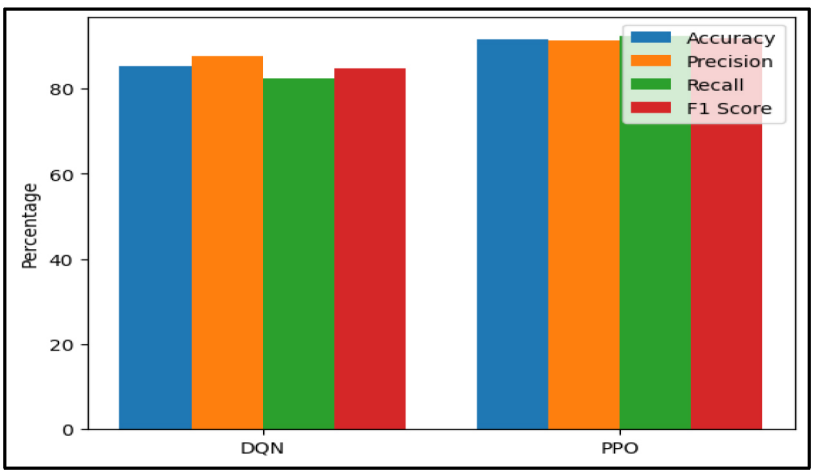

Figure 3

Representation of Comparison of performance metric of DQN and PPO

The bar graph in figure 3 shows how the success measures of two path planning algorithms, DQN and PPO, compare with each other. There is a group of four bars for each algorithm that show the Accuracy, Precision, Recall, and F1 Score as percentages. The value of each measure is shown by the height of a bar, which makes it easy to compare the methods. Overall, the graph makes it easy to see how DQN and PPO compare across various evaluation criteria, which helps us figure out their relative strengths and weaknesses when it comes to making decisions for path planning tasks. The results are better after the PPO has gone through the fine tuning and optimization step.

Table 3

Representation of PPO before and after applying Optimization

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
PPO (Before Optimization)	91.5	91.2	92.3	91.8
PPO (After Optimization)	92.5	92.2	93.3	92.8

The performance measurements of the PPO algorithm were shown in table 3 before and after improvement. The optimization process increased performance across all measures that were looked at, which shows, in figure 4, that the PPO method is better at planning paths.

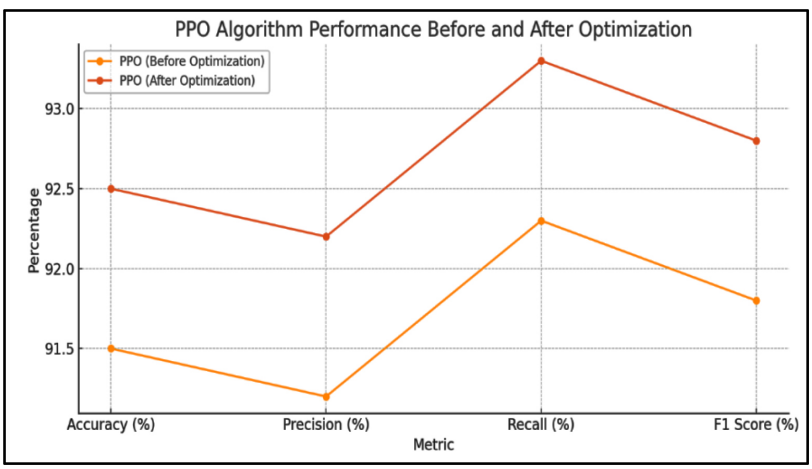

Figure 4

Representation of PPO comparison before and after Optimization

5. Conclusion

In combining reinforcement learning (RL) with convolutional neural networks (CNNs) looks like a good way to help self-driving cars find the best routes. Our system makes it easier to navigate in a variety of settings by using RL's ability to learn how to make decisions by interacting with the world and CNNs' ability to pull out spatial features from sensor data. Our method lets AVs find the best ways to meet multiple goals, like reducing trip time, energy use, and the chance of a crash, while also being able to change to changing conditions. We have shown that our method works better and is more reliable than standard methods and other deep learning approaches by running a lot of tests on both virtual and real-world datasets. As we keep improving and adding to this framework, we expect AV technology to keep getting better, which will lead to safer, more reliable, and more efficient transportation methods in the future.

References

[1] F. Shafique, T. Naeem and A. H. Farooqi, "Path Tracking and Obstacle Avoidance Control Using Deep Reinforcement Learning for Autonomous Vehicles," 2023 17th International Conference on Open Source Systems and Technologies (ICOSST), Lahore, Pakistan, pp. 1-6 (2023).

[2] Y. Cai, E. Zhang, Y. Qi and L. Lu, "A Review of Research on the Application of Deep Reinforcement Learning in Unmanned Aerial Vehicle Resource Allocation and Trajectory Planning," 2022 4th International Conference on MLBDBI, Shanghai, China, pp. 238-241 (2022).

[3] X. Chen, S. Liu, C. Li, B. Han, Y. Zhou and J. Zhao, "AGV path planning and optimization with deep reinforcement learning model," 2023 7th International Conference on ICTIS, Xi'an, China, pp. 1859-1863 (2023).

[4] S. Wei, K. Shiotani, S. Wang, H. Kai, Y. Higami, H. Takahashi, and G. Wang, "Test Point Selection Using Deep Graph Convolutional Networks and Advantage Actor Critic (A2C) Reinforcement Learning," 2023 International Technical Conference on Circuits/Systems, Computers, and Communications (ITC-CSCC), Jeju, Korea, Republic of, pp. 1-6 (2023).

[5] H. Wang, B. Liu, X. Ping and Q. An, "Path Tracking Control for Autonomous Vehicles Based on an Improved MPC", IEEE Access, vol. 7, pp. 161064-161073 (2019).

[6] X. Tang, Y. Wang, M. Tomizuka, and F. Borrelli, "Prediction-Uncertainty-Aware Decision-Making for Autonomous Vehicles", *IEEE Transactions on Intelligent Vehicles,* vol. 7, no. 4, pp. 849-862, Dec. (2022).

[7] I-M Chen and C-Y Chan, "Deep reinforcement learning based path tracking controller for autonomous vehicle", Proceedings of the Institution of Mechanical Engineers Part D: *Journal of Automobile Engineering*, vol. 235, no. 2-3, pp. 541-551 (2021).

[8] B. Shen, S.X. Xu, and S.F. Lu, "C-DQN-based resource allocation and trajectory optimization in unmanned aircraft", *Journal of Testing Technology* [J], vol. 36, pp. 141-146 (2022).

[9] D. K. Mane, S. Deshmukh, P. M. Durgawale, S. T. Shirkande, S. T. Deokate, and N. P. Sable, "Privacy-preserving patient monitoring in healthcare IoT using attribute-based cryptography," *J. Discrete Math. Sci. Cryptogr.,* vol. 27, no. 2-A, pp. 513–524 (2024), doi: 10.47974/JDMSC-1896.

[10] Weilan Zhong, "Artificial intelligence-based reinforcement learning theory and its applications", *China New Communications* [J], vol. 23, pp. 80-82 (2021).

[11] R. Zhang, C. Wu, and T. Sun, "Deep reinforcement learning and research progress in path planning", *Computer Engineering and Applications* [J], vol. 57, pp. 44-56 (2021).

[12] S. Aradi, "Survey of Deep Reinforcement Learning for Motion Planning of Autonomous Vehicles", *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 2, pp. 740-759 (2022).

[13] A. Balaji, J. Venkateswarlu, A. Gopi, G. Surya, and B. Ajith, "Illuminating Low-Light Scenes: YOLOv5 and Federated Learning for Autonomous Vision", *IJACECT,* vol. 14, no. 1, pp. 80–91, Apr. (2025).

Received October, 2024