

Adaptive learning-based navigation and obstacle avoidance for autonomous robots in unstructured environments

Ganesh Babu *Loganathan*^{1*}, Bercy Miraclean *Johnson Jebaraj*¹, Dilip Kumar *Sakthivel*¹, Harshitha *Ramamoorthy*¹, Smirithi Shree *Parasuraman*¹, Gokul *Saravanan*¹, Sundara Pandian *Paramasivam*¹, and Selva Lakshmi *Kannan*¹

¹Department of Robotics and Automation, Rajalakshmi Engineering College, Chennai 602105, Tamil Nadu, India

Abstract. Among the many fascinating and useful aspects of AI, reinforcement learning performs well. Using the principles of action and reward, reinforcement learning makes it easier to learn new tasks. The issue of robot navigation is tackled by motion planning. The ability to automatically react in real-time to changes in the environment is currently missing from motion planning methods. An intricate setting full of impediments exacerbates the situation. As a result of the capabilities of the reward system and feedback to the environment, robotic systems can be enhanced through reinforcement learning. Managing a complicated setting may get easier by using this. Current path planning algorithms converge to a solution late because they are computationally expensive, less responsive to the environment, and slow. Additionally, because of the need for post-processing, they are not as effective for task learning. The problem-solving capabilities of reinforcement learning lie in its action feedback and reward policies. This study introduces a new reinforcement algorithm that combines deep learning with Q-learning. The suggested method is tested in a space with limited space and a lot of obstacles. Additionally, we handle ways to improve the merging of collision avoidance and motion planning based on reinforcement learning. At the 640th and 690th episodes in a crowded and a small route environment, the agent of the suggested method converged. Based on the amount of turns and the planner's ability to converge the path, a state-of-the-art comparison reveals that the suggested strategy beat existing alternatives.

1 Introduction

Motion planning were process of determining best route for robots to take, when moving from one location to another.

*Corresponding Author: ganeshbabu.l@rajalakshmi.edu.in

Autonomous operation of mobile robots including those engaged in unmanned pursuit, rescue, agricultural, alien roving, and medical assistive robots relies on this crucial and fundamental quality. To make mobile robots operate autonomously, motion planning is essential. This problem has been solved using multiple planner approaches. For smaller-scale issues, grid-based methods have been employed, for example, A*, D*, and MEA* (Memory Efficient A*). However, for complicated problems with many dimensions, sampling-based techniques like RRT and RRT*-AB demonstrated outstanding performance. Also, path planning algorithms as ACO (Ant Colony Optimization), PSO (Particle Swarm Optimization) and GA (Genetic Algorithm) have been enhanced using heuristic-based evolutionary planners. The service industry is growing at a far faster rate than traditional manufacturing. A highly adaptable system that can keep energy consumption low is essential in today's industrial environments, especially in the age of Industry 4.0. The capacity to handle various environmental variables is particularly important in the motion planning area. It is computationally and energetically costly for a robot to refresh their sensors to acquire new information, which were necessary for its mobility plan. Feedback from the environment and human contact have recently been the basis of a great deal of effort. But in real-time navigation settings, existing motion planning methods cannot handle automated responses to the environment fast enough. Additionally, traditional path planning approaches like heuristic-evolutionary planners, grid-based approaches, or sampling-based approaches are computationally costly, less responsive to the environment and slow. In addition, owing to an inefficient path, post-processing is often necessary. Reinforcement learning's (RL) feedback capability potential has lately attracted attention in the motion planning sector as a means to address these limitations. Soft growing robots inspired by plant movement, are designed to navigate tight spaces and hazardous terrain. The study introduces a deep reinforcement Q-learning algorithm for enabling effective navigation of these robots in cluttered environments [1]. RL's action feedback and reward mechanisms make it quick to fix these problems. Agents, environments, reward signals, policies, and value functions are the typical key components of RL. The same way that people learn and adjust to different situations, its learning framework works. The agent is not given concrete instructions about how to act in every given situation; instead, it is left to its own devices to figure out what is expected of it through trial and error. A scalar reward is provided to the learning agent after each state update; the agent strives to maximize the return over time as the reward increases in relation to the quality of the transfer. RL's long-term return maximization and trial-and-error nature are two of its important qualities. However, recent advances in artificial intelligence and machine learning including autonomous robotics, self-driving cars, and image recognition systems are built around deep neural networks (DNNs). The remarkable accuracy and self-learning capabilities of deep learning methods have contributed to their meteoric rise in popularity. Novel approaches to the issue of motion planning have emerged as a result of recent developments in RL that make use of deep learning. Nevertheless, delayed convergence is a problem with these algorithms. Regarding this matter, the following are the contributions made by this study. One reason for deep learning technologies' skyrocketing popularity is their incredible accuracy and ability to learn on their own. The application of deep learning in RL has recently led to novel approaches to problem of motion planning. One issue with the algorithms is that they have delayed convergence. That being said, this study does contribute in the following ways.

- We provide a deep reinforcement learning approach to motion planning at complex environment with several obstacles. Using Q for motion planning, suggested method were simplified deep reinforcement learning strategy [2].
- An environment with tight congested corridors and another with structural impediments is used to test the suggested method.

- Presented a more convergent approach that uses less energy and has fewer turns, leading to a more efficient path with less path following time and less energy consumption overall.

2 Related Works

Improvements in the field of path planning for commercial and residential robots have been made possible by recent advances in learning-based methodologies. Field of mobile robot continues to expand path planning as fundamental yet challenging task. Traditional path method performs poorly in uncertain environments. A novel Deep Spiking Q-Network algorithm, which integrates global and local information, enhancing path planning performance [3]. Optimizing robotic arm path planning and control is critical for precision and efficiency. An innovative method to integrate the deep Q-learning (DQL) and reinforcement learning to solve the problems such as collision avoidance and dynamic control. The simulated results reveal that the performance has been greatly improved with a 98.76% index of accuracy in path optimization and minimized the computational expenses by 22.4%. [4]. Snake robots which can replicate the movement patterns of animals can be used to explore areas that regular robots cannot reach. Our suggestion is to use autonomous obstacle avoidance technique, which is based on deep and SLAM reinforcement learning of snake robot. It has been demonstrated in experiments that the robot is capable of planning routes and avoiding obstacles in dynamical environments [5]. Multi-goal deep reinforcement framework to enhance needle insertion accuracy has been proposed in this research since it is critical in invasive surgeries. The framework is applied to quick re-plan and risk management with the help of universal distributional Q-learning (UDQL) and universal value function approximation. The high accuracy and robustness of the method in terms of insertion is confirmed by simulation and experimental results as opposed to the former methods [6]. In this paper, the proposed deep reinforcement learning algorithm is a better improvement on the existing algorithm to overcome low learning efficiency and slow convergence of robot path planning. It uses Dueling DQN, a priority experience replays strategy and Artificial Potential Field (APF) algorithm to speed up convergence and improve exploration. The technique is more efficient in terms of learning speed, path planning and convergence rate when compared with the conventional algorithms [7]. In this paper, I propose a generalized coverage path planning algorithm based on the deep reinforcement learning methods. It has adopted an observation space with varying map sizes, an action masking scheme, and a special purpose rewarding scheme to make sure it is robust and adaptable. The framework has almost optimal performance in zero-shot learning and outperforms state-of-the-art algorithms in a variety of situations, which demonstrates its ability to achieve generalizable and versatile path-planning solutions [8]. The modified Deep Deterministic Policy Gradient algorithm is used to enhance path planning of unmanned robots at unknown environments. The process separates experience pool and a guided reward function is applied to promote obstacle avoidance and increased convergence. The outcomes of the simulation and real-life experiments indicate that the algorithm works quite well in the complicated, unknown environment, with higher target point localization [9]. Path planning in dynamic multi-agent setting is a complex task due to agent-obstacle interactions. The current paper presents a Deep Q-Network (D3QN) that is a Double Dueling network employed in collaborative path planning that includes multi-agent positional constraints and CNN-LSTM sensor fusion network. The results of the simulation indicate that D3QN algorithm is superior to deep learning and it can be chosen in situations when a rapid convergence is needed, as well as it has a greater success rate under the conditions of dynamic character. [10].

Deep reinforcement learning and Deep Deterministic Policy Gradient method of a large-scale path-planning algorithm of underwater robots. This model is developed with the help of TensorFlow to plan the path of the robot by uniting exploration and exploitation strategies. Experimental findings indicate that the algorithm is useful towards the enhancement of accuracy and efficiency in underwater navigation [11]. This paper proposes a decentralized path planning system of multirobot that uses deep reinforcement learning, namely Asynchronous Multi-Critic Twin Delayed Deep Deterministic Policy Gradient method. The strategy does not require any shared state and intention information, increasing scalability and flexibility. The outcomes of the simulation indicate that the model is superior to the conventional centralized approaches to dynamical conditions with a different population of robots [12]. A new algorithm named KAI-DDPG that incorporated the use of both the kinematics analysis and the immune optimization to enhance the efficiency of the DDPG algorithm in path planning. The approach will maximize the reward feature as it includes various performance measures including the orientation angle, linear velocity, and safety. The findings show that KAI-DDPG can solve the negative aspects of the use of traditional DDPG as it provides better training efficiency and application [13]. A better Double DQN (DDQN) path planning algorithm is suggested to overcome the problem of dimensionality, model convergence, and sparse rewards. The approach combines the dimensionality size reduction and expertise knowledge to optimize the reward statistic and speed up training. It has been experimentally demonstrated that the suggested approach enhances convergence, stability, and path quality that lead to smoother and shorter paths in the virtual and real-life scenarios [14]. A footstep tracking algorithm based on deep reinforcement learning to allow robots to move in dynamic complex 3D settings. The strategy takes into consideration periodic and symmetrical information in the rewarding program. This is evidenced by the experimental outcomes which reveal that reinforcement learning supplemented by real time footstep planning shortens training and eliminates redundant path-planning knowledge [15]. The paper describes a local planning technique using a deep reinforcement learning model known as RND3QN that is an integration of reward-based exploration and n-step dueling double DQN algorithms. The approach enhances the navigation by minimizing bias and adding auxiliary reward to increase reward allocation in sparse spaces. The results of simulation proves that the proposed method is more efficient with a success rate doubling to 174 percent in comparison to D3QN. The Deep Q-Network (DQN) model of the global path planning of Automatic Guided Vehicles (AGVs) is suggested in order to optimize a number of starting and end points. The model enhances convergence and generalization through manipulating the input state and reward functions where as well as integration of alternative AGV states in the course of training. The algorithm is better than a A* and RRT algorithm in a multi-target environment as shown by the simulation results. In this paper, a path planning strategy of manipulator in dynamic settings is presented based on the Soft Actor-Critic (SAC) algorithm. The approach involves extensive reward of target approach and active obstacle deterrence. Findings exhibit better sample use and performance in real time obstacle avoidance as the simulation experiments indicate. The article discusses the optimization of self-planning paths of mobile robots in interior decoration based on Proximal Policy Optimization (PPO) algorithm. It introduces an adaptive loss function and a Credit Assignment Problem (CAP) model to improve path planning and learning efficiency. The outcomes indicate improvements at path planning success rates with good adaptability to dynamic environments. Deep reinforcement learning-based path planning for picking robots at dynamic environments is proposed. The method utilizes a directional penalty obstacle avoidance function and a reward system based on target attraction and obstacle rejection. Simulation results show a 97.5% success rate in picking tasks, outperforming traditional methods in both obstacle avoidance and planning efficiency.

3 Proposed Approach

Reinforcement learning is taken into account, the difficulty of goal-directed agent dealing with uncertain environment.

The agents with an unpredictable environment are typically modeled by Markov decision process (MDP). The typical parts of a Markov decision process (MDP) include state space S , action space A , reward space R , transition probability, and the discount factor. The one that was released employs RL and contains a less advanced DNN, which is different from the existing approaches. These assumptions informed the development of proposed deep reinforcement learning technique.

- The quantity of hazards in setting should be limited.
- The setting is secluded.
- It is imperative that, the robot can detect any and all nearby impediments.
- Everything is fine with the localization sensors.
- What follows is an explanation of the basic ideas behind the proposed technique and the whole procedure.

3.1 Preliminaries

- **The agent.** The component responsible for making decisions in RL is known as the agent. In making its choice, the agent is unrestricted in its use of both internal rules and environmental observations.
- **The reward.** RL, the agent is rewarded with a scalar value after carrying out action. Agent receives reward of 1 value if they reach the target; if they encounter obstacles or fail to reach the target location, they receive a reward of -1 value.
- **Actions.** It is possible to classify the activities in RL-based algorithms as continuous or discrete. An agent's position can change in a discrete action by going up, down, left, or right, but in a continuous action, the value of the action remains constant. In RL, the agent-environment interaction is illustrated in Fig. 1. As a result of the agent's actions, the environment responds by rewarding them and providing them with information about the future stage.

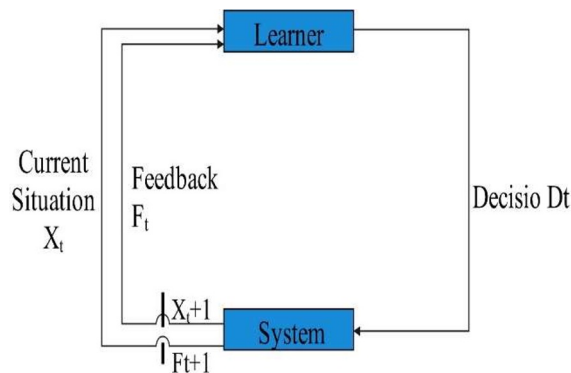


Fig. 1. Reinforcement Learning Framework: Agent-Environment Interaction

3.2 Q-Learning

When it comes to reinforcement learning algorithms, Q-learning is among the most fundamental. Even most basic Q learning could point grid world in right direction by solving state problem and discrete action. The safe navigation of mobile robots amidst dynamic obstacles is a critical challenge. Deep Q-Network (DQN) algorithm with inflated robot reward functions for collision-free path planning. It gives a Q table, where each state is represented by a row and the actions performed at that state are represented by columns. At state s , when an action is completed, grid, Q, represents discounted reward. Agent should respond in line with the greatest value at all column if data in the table resembles reality. The Q table needs to be started with a fixed value begin. Agent then makes a decision time t , shows a reward at time r_t , moves on to state s_{t+1} , and modifies Q. The procedure is a direct value iteration update as shown in Eq 1.

$$Q^{\text{new}}(s_t, a_t) = (1 - \alpha)Q^{\text{old}}(s_t, a_t) + \alpha(r_t + \gamma \max_a Q(s_{t+1}, a)) \quad (1)$$

$Q^{\text{new}}(s_t, a_t)$: In state y -Z., this stands for the revised Q-value for action a -Z.

- $Q^{\text{old}}(s_t, a_t)$: Prior to the change, this was current Q-value for action at state y -Z.
- α : The pace of learning, which controls the extent to which fresh data supersedes previously stored data. Between zero and one is its range. More weight is given to new information when α is higher
- r_t : The benefit gained from an action in a state where it is performed.
- γ : The weight of future benefits is determined by the discount factor. Between zero and one, it falls. Future rewards are more substantial with a greater γ .
- $\max_a Q(s_{t+1}, a)$: The highest Q-value, out of all conceivable actions a , for the following state b -Z+1. In terms of future rewards, this is the best that next state can offer.

Explanation of equation. The calculation takes into account both the previous Q-value and the new value, which is sum of reward and discounted maximum Q-value for future. Presented below is a detailed explanation:

Current Q-value:

$$(1 - \alpha)Q^{\text{old}}(s_t, a_t) \quad (2)$$

- A part of the current Q-value is retained by this term.
- As β grows, the impact of the previous Q-value is diminished by the factor $(1-\alpha)$.

Learned Value (New Information):

$$\alpha(r_t + \gamma \max_a Q(s_{t+1}, a)) \quad (3)$$

- The immediate benefit of the activity is represented by the reward, a_b .
- The largest anticipated future reward from next state, y_{t+1} , before discounting by γ is denoted as $\gamma \max_a Q, y_{t+1}, a$

Update Rule:

- Q-value, Q -new yz, a_t an amalgamation of previous Q-value and updated data.
- The ratio of old to new knowledge is controlled by the learning rate α . Update is more dependent on existing Q-value when β is near to 0 and on fresh information when it is close to 1.

3.3 Environment Formulation

In the research, motion planning issue were modeled as Markov decision process, denoted as combination (S, A, T, R), here S is state space of the system and A is its action space. The procedure moves action $\epsilon \in A$ from one entity to another at each time stage. The probability that function will remain at state S following the execution of action were $P(s' | s, a) \cdot R: S \times A \rightarrow R$. The $R: S \times A \rightarrow R$ uses reward function R.

The mechanism receives reward with monetary value at each time step based on its current state and its actions. The performance of current rewards relative to future ones is reflected by the $\gamma \in [0,1]$ discount factor. For each state $e \in S$, the agent's $\pi: S \rightarrow A$ describes action $a \in A$. By definition, a policy is optimum if and only if it maximizes the expected return from the starting point. When policy π is continually followed, value function $a\pi(y)$ were defined as an anticipated return starting states, i.e., $S=y$.

Although $v^\times(s)$ were defined as maximum acceptable value of $a\sigma(y)$. Specifically, state s , action a , and policy σ , action value of pair (s, a) under σ were $q\sigma(s, a)$, and stellar action-value function were referred to as $q^\times(s, a)$. In model-based setting, MDP problem can be solved using dynamic programming techniques if the probability of state transition (P). Since the likelihood of a transition is not known in a model-free setting, RL works well there. The approach is tested in simulation and real-world environments, demonstrating higher success rates and optimal task completion compared to other DRL algorithms.

3.4 Action Selection

There are four potential acts that the mobile robot can do. Robot can direct to backward, forward, left, or right. Whether it travels forward, left, right, or backward, the robot can only do one action at a time b .

3.5 Reward Function Selection

Eq. 2 shows the design of reward function for mobile robot.

$$R = \begin{cases} 1, & \text{Reach target} \\ -1, & \text{Encountering obstacles} \\ 0, & \text{Otherwise} \end{cases} \quad (4)$$

A positive reward with a value of one is given to mobile robot if it reaches target. A negative reward of -1 is given to the mobile robot if it crashes into an obstacle, whereas in all other cases the reward is zero. Reinforcement learning enhances task learning through action and reward principles. Motion planning for robots often lacks responsiveness in complex environments. Q-learning-based reinforcement algorithm, evaluated in narrow and cluttered passage environments, outperforms existing methods in terms of turns and path convergence.

3.6 Step Function

We must transfer the issue into the DRL framework to integrate robot motion planning with DRL. The agent's interaction with its surroundings is illustrated in Fig. 2. Thus, an agent-environment interaction was created. We shall begin with essential functionalities to comprehend the DRL configuration. Actually, the rules for the agent's interaction with its surroundings are determined by the render (), reset (), and step () functions. To clear the DRL environment and see how it looks, the render () and reset () methods were used. Initiating communication with its environment, the agent chose action A. The agent finds out, using the Boolean flag, whether there is a collision with barriers on a specific motion or not. If true or yes, environment will reset and returning values are the current reward. If it is not true, it will construct a new target position after the agent has already moved to it.

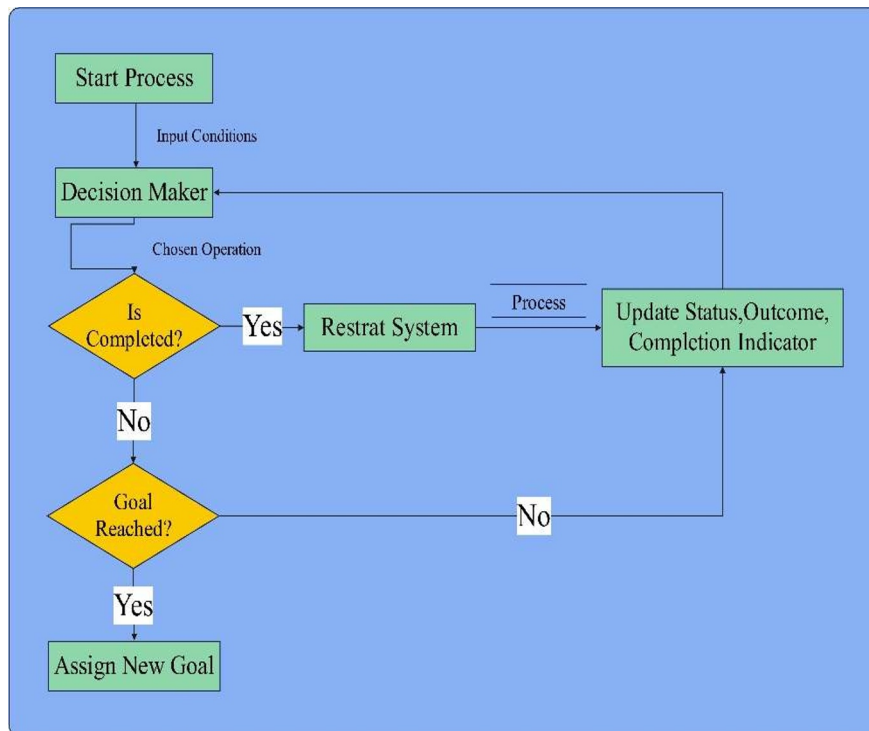


Fig. 2. Process Flow of step () Function

3.7 Detailed Architecture of Deep Reinforcement Q-Learning

An all-inclusive deep reinforcement Q-learning framework using the neural network to approximatively define Q-value function $Q(y, a)$, the DQN framework estimates the projected cumulative reward a at state were no separate neural network representing the policy, unlike in an Actor Critic framework. Alternatively, the defined implicitly by Q-value estimations, with an epsilon-greedy strategy for action selection: $\rho(s) = \operatorname{argmax}_a Q(y, z)$. Fig. 3 shows that the network consists of two full layers, with a total of 128 neurons in each layer. Input consists of two-dimensional state data. The suggested method takes this two-dimensional data and turns it into a numerically measurable model that the neural network can use for training. The total design becomes very lightweight for testing and training thanks to this simple pre-processing phase. The input consists of two-dimensional state data. The agent's action value, which it utilized to progress to the subsequent state is the output. In order to activate the buried layers, ReLU function were utilized. Adam utilized loss function, which is a mean square error as an optimizer function.

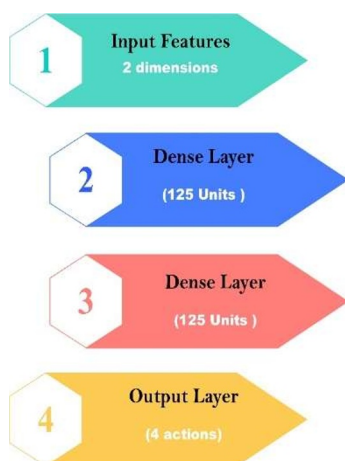


Fig. 3. Network architecture

There is a suggested DQN shown in Fig. 4. We forecasted the optimal Q-values using feed-forward neural network in suggested approach. The agent needs to remember past events for this method to function. In order train network, it makes use of previously stored experience. The idea of experience replay is used by deep Q learning to enhance performance. The training process is stabilized by employing this concept.

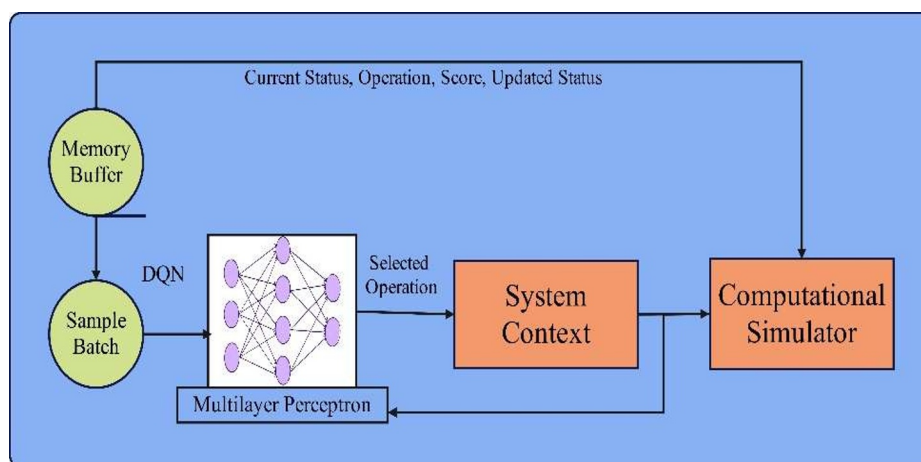


Fig. 4. Data Flow and Key Elements of the Proposed Algorithm

Just like any other memory, experience replay stores previous events as tuple (y, y', a, r) , where y shows current state, y' is the action, and r reward. Many objectives of Q learning are network with environmental knowledge. Once the agent has chosen an action at random and taken the epsilon value into account, they will carry it out, watch for the reward r , and then proceed to the next state, y' . The experience replays memory b, y', a, r stores this data. In order to train the Q-Network, a selection of batches at random was taken from the experience replay memory. Table 1, shows the neural network's summary. There are 32,644 total parameters that can be trained in the neural network.

Table 1. Summary of neural network architecture

No. of Parameters	Output Shape	Layer (Type)
0	(121)	flatten (Flatten)
15616	(128)	dense (Dense)
16512	(128)	dense_1 (Dense)
516	(None, 4)	dense_2 (Dense)

method 1 presents the pseudo-code for the suggested method that incorporates experience replay.

Algorithm 1 A New Method for Robot Path Planning in Unknown Environments Based on Deep Reinforcement Learning to Avoid Collisions.

- Input: Locations of the environment's source and target
- Initialize: Memory ability for reliving experiences D up to N
- Initialize: Random weights θ applied to the action function Q
- Initialize: Weights $\theta^- = \theta$, the target action function Q^-
- Set: Rate of exploration $\epsilon \in [0, 1]$, rate of decay, minimum per unit time \min
- Set: Learning rate α and discount factor $\gamma \in [0, 1]$ are both ranges from 0 to 1.
- from the first episode to the last, do
- Get the initial state s_1 and set up the environment.
- throughout the time interval from 1 to T , perform
- Pick an arbitrary course of action a_t
- Alternatively, choose $a_t = \operatorname{argmax}_a Q(s_t, a; \theta)$.
- Go to state s_{t+1} after you have executed the action a_t and seen the prize r_t .
- Put the sequence (s_t, a_t, r_t, s_{t+1}) into D for storage.
- Minibatch of transitions (s_j, a_j, r_j, s_{j+1}) from D , sampled at random
- Compute:
 - $y_j = \begin{cases} r_j & \text{if terminal state} \\ r_j + \gamma \max_a \hat{Q}(s_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$
- With regard to network parameters θ , execute gradient descent step on the square of $(y_j - Q(s_j, a_j; \theta))$.
- C steps, $Q^- \leftarrow \frac{1}{4} Q$ (i.e., $\theta^- = \theta$)
- Change s_t to s_{t+1} .
- Bring it down to minimum by decaying
- When target state was attained or maximum number of steps have been taken
- Season finale
- terminate when
- conclude after

4 Results and Discussions

A personal computer is used to evaluate the algorithms that are written using Python 3. A 64-bit version of Windows 10 is installed. From benchmark data sets, two distinct case studies of environment maps were taken into consideration. In order to test suggested method's efficacy, a software simulator of a mobile robot agent operating in a two-dimensional environment is developed. The openAI gym framework is used to construct the 2-D environment.

4.1 Case Studies Environment

An indoor setting is shown in the simulation experiments. The world in question is a 9 by 9 grid. In scenario M1, there are many difficulties in the environment. Fig. 5 shows the agent's target position marked in M1, with its initial position at upper left corner.

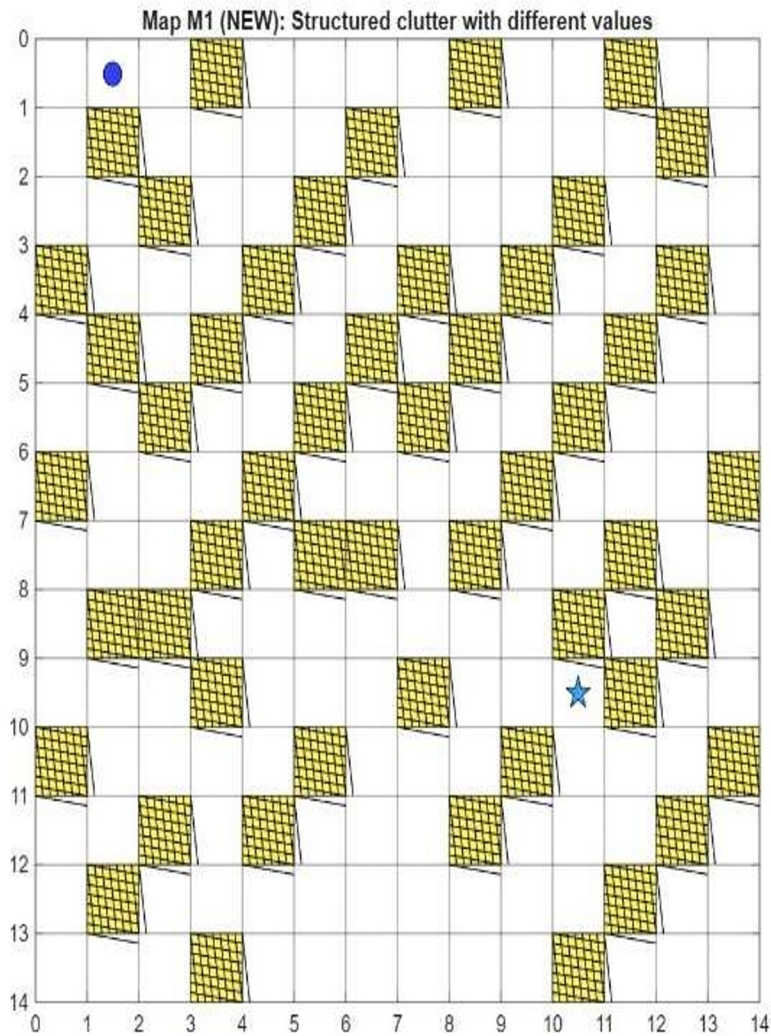


Fig. 5. Map M1: Obstacle-Rich Environment with Structured Layout

The size of the occupancy grid in M2, a convoluted and narrow route is 11×11 . Based on the information presented in Fig. 6, the M2 environment agent starts in the upper left corner and needs to move to the bottom right corner to reach its destination.

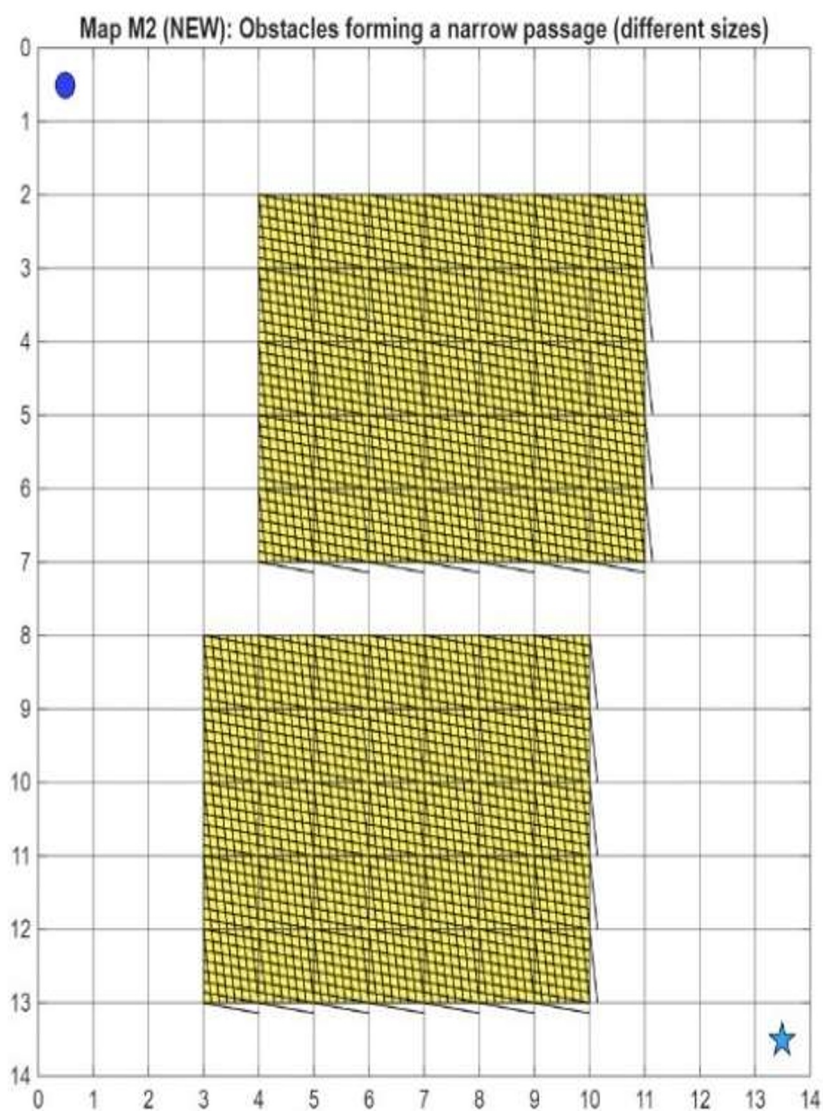


Fig. 6. Map M2: Obstructed Environment Creating a Narrow Pathway

According to the environment maps, mobile robot starts out in top left corner. Occupied grid cells on the environment map represent impediments in this grid-based setting. Finding the route from its current location to the flagged destination is the objective of the mobile agent in this environment. As the white grid cells indicate, the mobile agent is free to move in any open location. The agent has no idea what the surroundings are like.

4.2 Evaluation Measurement

In order to measure success of suggested strategy, we employ following evaluation criteria:

- How many times the agent was able to find the path to the target point is indicated by the number of convergence episodes. We prefer convergence with fewer episodes.

- Also, depending on the collision avoidance success rate and the determined location score, an episode can have more steps yet still be better than another. Therefore, the success rate is the determining factor in whether an episode is good, rather than the number of stages.
- A robot's number of turns indicates the total number of turns it takes to go to a specific location. For the purpose of motion planning, it is a crucial metric. As the number of turns increases, the robot will need to slow down at each turn, change its angle and direction, and then speed up again. As a result, the robot will use more energy, have to work more, and spend more time following paths during application because of this entire procedure. The robot ages too quickly when its route includes unneeded turns.

4.3 Results

The results of suggested approach's experiments at two distinct settings are discussed in this section. From its starting point to the destination, the mobile robot navigates its environment to carry out motion planning. To determine the optimal values for the algorithm's hyperparameters, several experiments were carried out. To evaluate how various learning rates, exploration factors, and discount factors impact an algorithm's performance, simulations are run. We evaluated the suggested method on three fronts: path length, number of turns, and convergence episodes per iteration. Table 2 shows finalized parameters for suggested approach's success.

Table 2. Final parameter for proposed approach after trend analysis

Learning rate	Activation Function	Memory	Exploration factor	Batch Size	Discount Factor	Optimizer
0.01	ReLU	20000	0.9	32	0.95	ADAM

Results from various parameter settings for M1 and M2 are displayed at Tables 3 and 4. According to results, learning rate and exploration factor significantly affect the suggested method's performance. Following is a description of the effect parameters utilized in Tables 3 and 4:

Table 3. Impact of trend analysis parameter impact for M1

Learning Rate	Discount Factor	Exploration Factor	No. of Convergence Episodes	Maximum Steps	Minimum Steps
0.4	0.4	0.5	640	108	18
0.3	0.5	0.6	590	138	17
0.2	0.6	0.7	No convergence	315	16
0.1	0.7	0.8	No convergence	305	18
0.01	0.8	0.85	No convergence	510	15

- Discount rate reveals the significance of future benefits to the present situation. The algorithm converges as the discount rate is increased, as demonstrated in Tables 3 and 4.
- Learning rate is a trainable hyperparameter for neural networks that can be calibrated with a tiny positive value, often between 0.02 and 0.42.
- Exploration factor is an adjustable variable. As part of its exploration process, the agent made use of the exploration factor. When this value is close to 0, the agent does not investigate its surroundings, however when it is close to 1, it does so.

- The agent discovers the route to the destination as the exploration factor grows, as demonstrated in Tables 3 and 4.

Table 4. impact of trend analysis parameter's for M2

Discount Factor	No. of Convergence Episodes	Exploration Factor	Learning Rate	Maximum Steps	Minimum Steps
0.6	No convergence	0.62	0.02	640	21
0.7	No convergence	0.72	0.12	710	20
0.8	690	0.81	0.22	780	18
0.85	670	0.87	0.32	620	19
0.9	640	0.91	0.42	320	22

4.4 Comparison with Existing Approach

Fig. 7(a) depicts a structured, congested, obstacle-filled environment, and Map M1 is the corresponding scenario. Results were better with the suggested deep Q-learning compared to Q-learning algorithm. The plots in Fig. 7(b) clearly demonstrate that the prior Q-learning agent needed a significant number of episodes after discovering path to target at 300th episode. Nevertheless, as illustrated in Fig. 7, the aim might be reached in the 640th episode by utilizing the suggested approach agent.

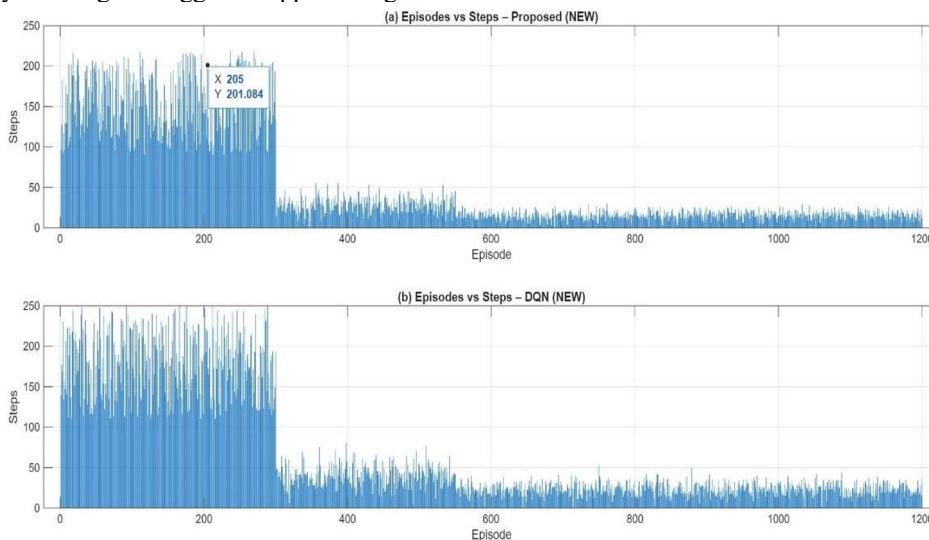


Fig. 7. Comparison of Episodes by Steps for M1: (a) Proposed, (b) DQN

For case M1 of the environment map, Fig. 8 displays the route planning of both the suggested method and the current Q-learning method. The simulation findings show that our method of robot motion planning outperforms the current method in M1, as agents learn more effective actions to reach the goal.

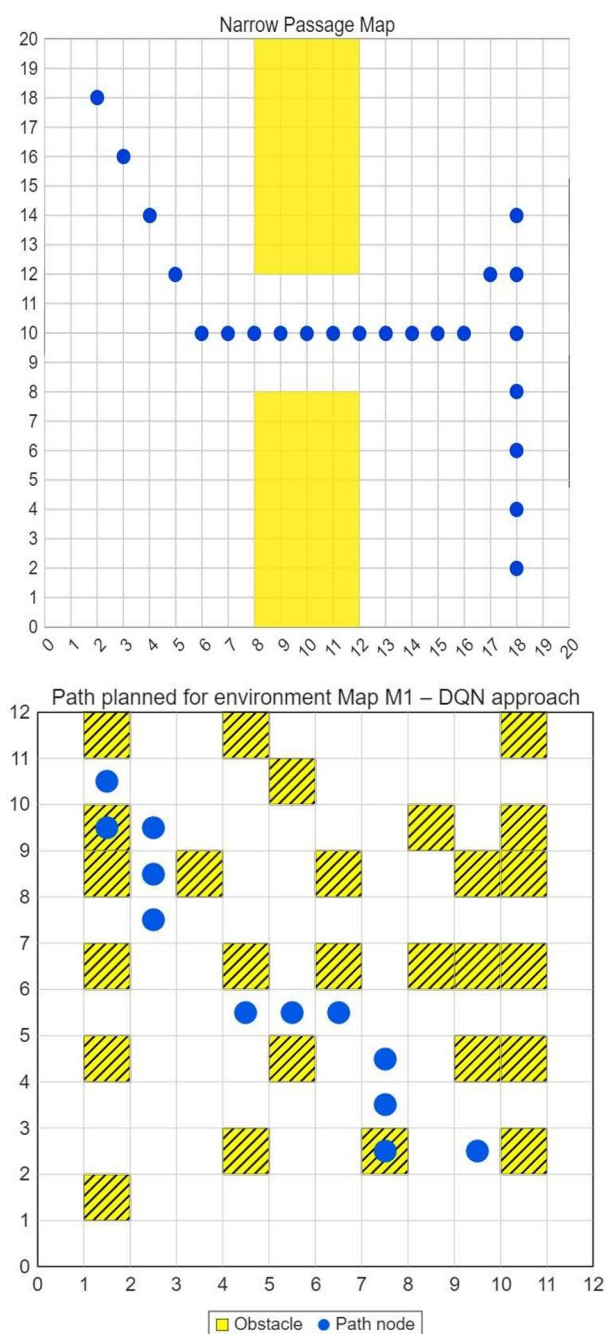


Fig. 8. Trajectory Planning in M1: Proposed & DQN

Results were better with the suggested deep Q-learning compared to Q-learning algorithm. Plots in Fig. 9 clearly demonstrate that the target was reached in the 650th episode using the previous Q-learning agent, and stabilization required a significant number of episodes later. But as seen in Fig. 9, the 690th event was when the target was located by means of the suggested approach agent.

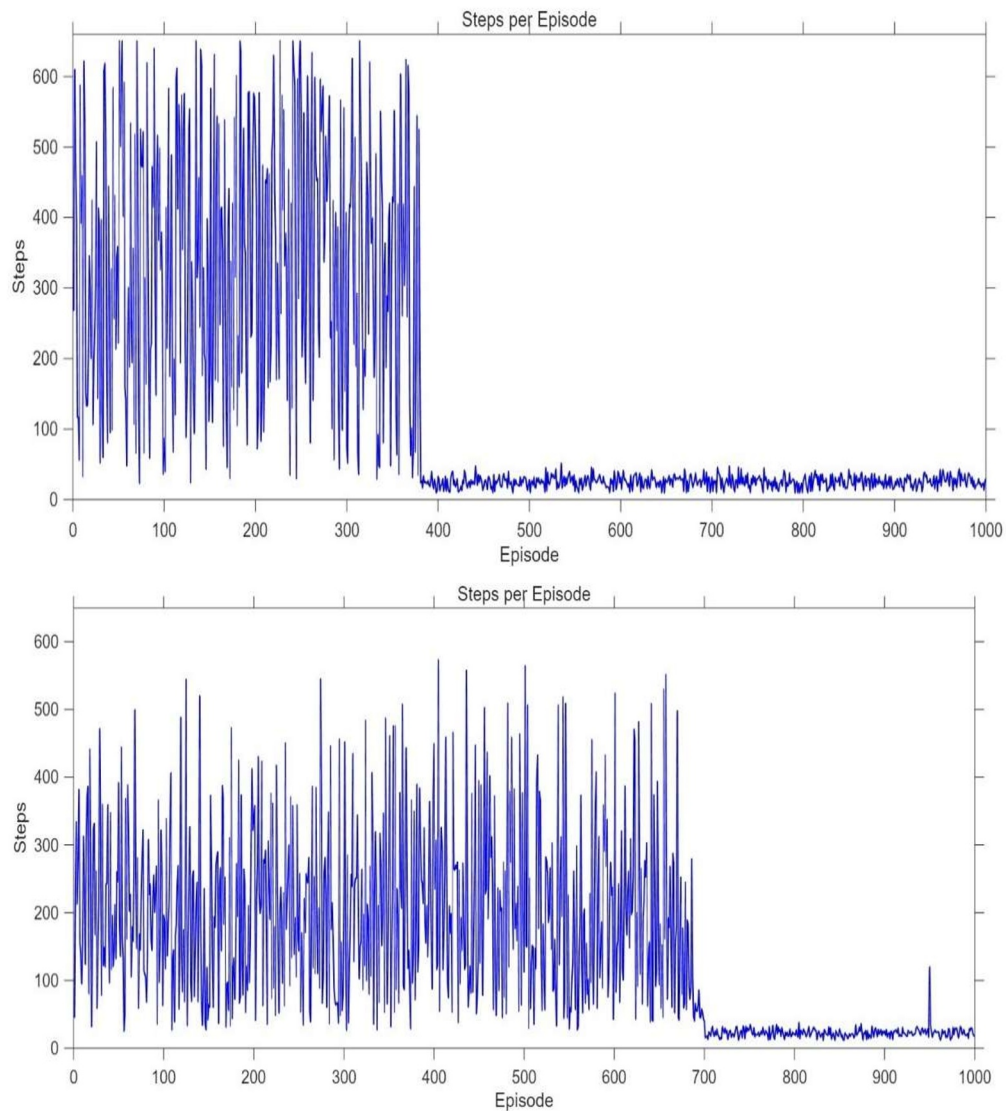


Fig. 9. Stepwise Episode Comparison in M2: Proposed & DQN

For a robot in M2, Figure 10 shows path planning for both existing method and suggested approach. When compared to the current Q-learning method in M2, the suggested method for mobile robot motion planning also performed better in the simulations. Additional deep reinforcement learning methods exist, however they are computationally and resource heavy, employ complicated deep neural networks in 3D mode of operation, and so on. In contrast, the suggested method produces a 2-D deep net that is reasonably lightweight and has much better convergence.

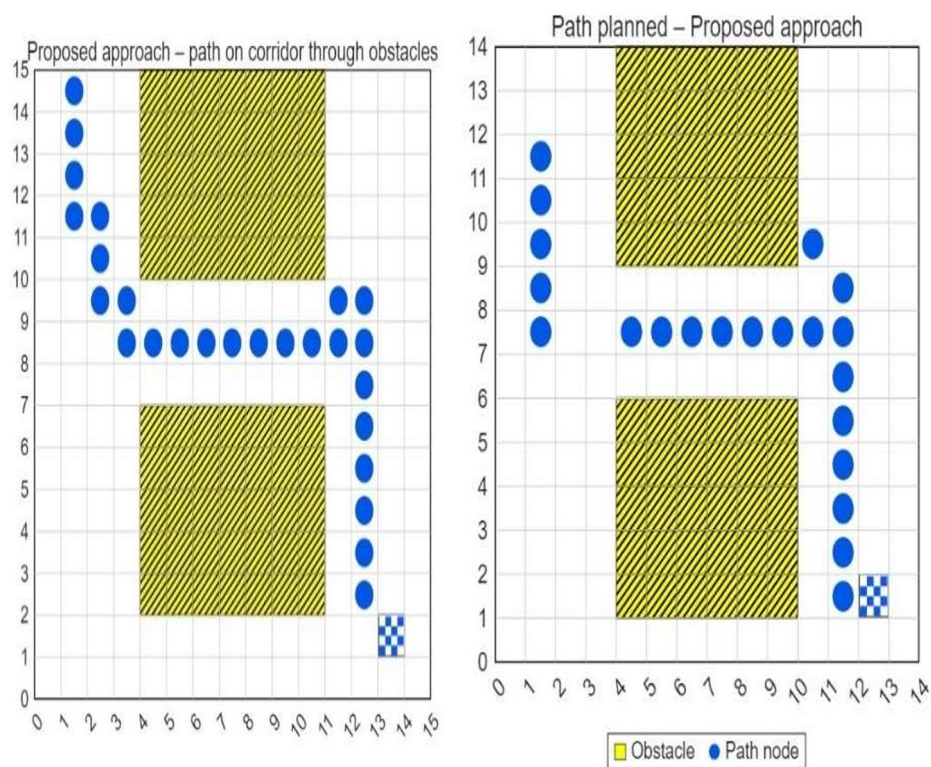


Fig. 10. Planned Paths in Environment Map M2: Proposed Method & DQN Method

Fig. 10 shows the proposed approach stacks up against the existing methodology using the M1 assessment matrices. The results show a comparison between the present strategy and our suggested one utilizing evaluation matrices for M1 and M2. An earlier version of Q-learning required the agent to maintain a table rows showing columns and states representing actions. Agent advances to the next state by earning rewards for all action. The system utilizes the database to record the benefits it obtains for different acts as it wanders around. There are noticeable improvements in convergence episodes and number of turns when comparing the suggested approach's performance to that of the classic DQN. In contrast to the conventional DQN, which needed 300 episodes and 7 turns to attain convergence for environment map M1, the suggested technique only needed 640 episodes and two turns. In the same vein, the conventional DQN takes 650 episodes and 7 turns to converge in environment map M2, while the suggested method takes 690 episodes and 6 turns. It is evident from these data that the suggested method guarantees faster convergence and uses less turns than the traditional DQN method. In place of the table, the suggested method has incorporated a deep neural network go to next stage. Neural network takes 2-dimensional array as input and returns an action size as output; agent selects action at random to the epsilon value, and then receives a reward and moves on to the next state. We use mini-batch and experience replay memory in our suggested method. Just like a memory saves the agent's experience, experience replay does the same. To train a neural network, we use the mini-batch method, which involves selecting random specimens from experience replay memory.

4.5 Comparative Significance of Proposed Algorithm over Traditional DQN

We emphasize the following points to address the novelty of the suggested algorithm in comparison to classic Deep Q-Network technique:

- **Simplified structure:** The strategy relies on a deep learning architecture that was initially developed to move through obstacles motion planning. This makes computation simpler and training and execution more efficient in comparison to conventional DQN that may be unsuitable in such jobs.
- **Enhanced Environment Handling:** The algorithm is tested in two challenging conditions, the first one is structural barriers, and the second is tight, crowded pathways. The standard testing environments of the classic DQN algorithm are simpler grid-based environments. The strength and flexibility of the suggested method is evidenced by the fact that it was tested under more complicated conditions.
- **Improved Convergence and Path Efficiency:** The suggested method streamlines the process of reaching an optimal policy by prioritizing the creation of straight pathways. This makes for most efficient and realistic solutions real-world robotic uses by reducing energy consumption and the time necessary to discover the path.
- **Experience Replay Optimization:** The experience replay mechanism is more effectively fine-tuned to the diverse and dynamic nature of the environment to create a stronger and more efficient process of learning. Such optimizations make the algorithm more generalizable and perform better than more traditional DQNs that use more basic experience replay.
- **Energy-Efficient Path Planning:** The proposed approach is different to the traditional DQN algorithms, as it aims at generating a less-turn heavy, less-consumption-intensive route. This breakthrough is even more significant in the given sphere because the energy efficiency of robotic systems directly relates to their operating life and usability.

5 Conclusions

To investigate the profound reinforcement learning algorithms, this study explores this topic in order to plan motion in a given environment. We offer a reinforcement learning method based on a deep neural network, which is not overly complicated. It is feed-forward in nature and takes input in the shape of a 2D array. The experiments are conducted in both static crowded and narrow path 2D environments which are obstacle filled. The effectiveness of the proposed approach is evaluated with the help of parameters like (0.95) discount factor (0.01) learning rate and (0.9) exploration factor. The proposed approach is better in terms of convergence rate and stability rates, as compared with the state-of-the-art approaches, based on the experimental results. The results show that it requires the agent 640 training standards on cluttered environment and 690 standards on restricted route environment to find a path based on the recommended strategy. Moreover, the proposed route is also energy saving compared to the previous mechanism since it has less possible turns. The next step that this research will take is to investigate the capability of robots to plan their movements in a three-dimensional space where there are moving impediments. The use of continuous action is another research possibility in this direction. Integrating the suggested reinforcement strategy with sampling-based methods, including such RRT*-AB, and subsequently extending it to cover an area issue would be a challenging part of the future research.

References

1. H. El-Hussieny, I. A. Hameed, Obstacle-aware navigation of soft growing robots via deep reinforcement learning. *IEEE Access*. **12**, 38192 (2024). <https://doi.org/10.1109/ACCESS.2024.3375340>
2. W. Zhao, Path planning of agricultural robots based on improved deep reinforcement learning algorithm. *Transactions of the Chinese Society of Agricultural Engineering*. **59**, 1492 (2025). <https://doi.org/10.3785/j.issn.1008-973X.2025.07.017>
3. A. Kumar, L. Zhang, H. Bilal, S. Wang, A.M. Shaikh, L. Bo, A. Rohra, A. Khalid, DSQN: Robust path planning of mobile robot based on deep spiking Q-network. *Neurocomputing*. **634**, 129916 (2025). <https://doi.org/10.1016/j.neucom.2025.129916>
4. K. Prabhavathi, S.R. Saratha, R. Malathy, R. Girimurugan, and K. Saranya, Mathematical foundations and computational techniques for robotic motion: a unified approach, Editor(s): S. Sountharajan, M. Karthiga, Balamurugan Balusamy, Ali Kashif Bashir, In *Medical Robots and Devices: New Developments and Advances, Applied Mathematical Modeling for Biomedical Robotics and Wearable Devices*, Academic Press, 37-58, 2026. <https://doi.org/10.1016/B978-0-443-33514-3.00009-5>
5. Y. Li, B.-W. Min, H. Liu, Deep Q-learning-based optimization of path planning and control in robotic arms for high-precision computational efficiency. *International Journal of Advanced Computer Science and Applications*. **16**, 1199 (2025). <https://doi.org/10.14569/IJACSA.2025.01601115>
6. X. Liu, S. Wen, Y. Hu, F. Han, H. Zhang, H. R. Karimi, an active SLAM with multi-sensor fusion for snake robots based on deep reinforcement learning. *Mechatronics*. **103**, 7823 (2024). <https://doi.org/10.1016/j.mechatronics.2024.103248>
7. S. Sachan, P. M. Pathak, addressing unpredictable movements of dynamic obstacles with deep reinforcement learning to ensure safe navigation for omni-wheeled mobile robot. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*. **239**, 1267 (2025). <https://doi.org/10.1177/09544062241281115>
8. Krishnamoorthy N., Muhammadu Sathik Raja, Suresh Kumar K., Ashika Preethi S., Chandraprakash R., Girimurugan R., Safety and stability analysis of robot systems, Editor(s): S. Sountharajan, M. Karthiga, Balamurugan Balusamy, Ali Kashif Bashir, In *Medical Robots and Devices: New Developments and Advances Applied Mathematical Modeling for Biomedical Robotics and Wearable Devices*, Academic Press, 157-173, 2026. <https://doi.org/10.1016/B978-0-443-33514-3.00016-2>
9. C. Li, X. Yue, Z. Liu, G. Ma, H. Zhang, Y. Zhou, J. Zhu, A modified dueling DQN algorithm for robot path planning incorporating priority experience replay and artificial potential fields. *Applied Intelligence*. **55**, 366 (2025). <https://doi.org/10.1007/s10489-024-06149-8>
10. J. P. Carvalho, A. P. Aguiar, Deep reinforcement learning for zero-shot coverage path planning with mobile robots. *IEEE/CAA Journal of Automatica Sinica*. **12**, 1594 (2025). <https://doi.org/10.1109/JAS.2024.125064>
11. L. Wei, Q. Xu, Z. Hu, Mobile robot path planning based on multi-experience pool deep deterministic policy gradient in unknown environment. *International Journal of Machine Learning and Cybernetics*. **15**, 5823 (2024). <https://doi.org/10.1007/s13042-024-02281-6>
12. S. Li, M. Li, Z. Jing, Multi-agent path planning method based on improved deep Q-network in dynamic environments. *Journal of Bionic Engineering*. **29**, 601 (2024). <https://doi.org/10.1007/s12204-024-2732-1>

13. V. Vimala, M. Manochitra, Muhammadu Sathik Raja, J. Sam Charles Devaprasad, R. Girmurugan, and B. Gayathri, Motion planning and trajectory optimization for robots, Editor(s): S. Sountharajan, M. Karthiga, Balamurugan Balusamy, Ali Kashif Bashir, In *Medical Robots and Devices: New Developments and Advances*, Applied Mathematical Modeling for Biomedical Robotics and Wearable Devices, Academic Press, 135-155, 2026. <https://doi.org/10.1016/B978-0-443-33514-3.00014-9>
14. S. Wang, S. Piao, X. Leng, Z. He, Learning 3D bipedal walking with planned footsteps and Fourier series periodic gait planning. *Sensors*. **23**, 1873 (2023). <https://doi.org/10.3390/s23041873>
15. H. Yin, C. Wang, C. Yan, X. Xiang, B. Cai, C. Wei, Deep reinforcement learning with multicritic TD3 for decentralized multirobot path planning. *IEEE Transactions on Cognitive and Developmental Systems*. **16**, 1233(2024). <https://doi.org/10.1109/TCDS.2024.3368055>