Journal of Computer and Knowledge Engineering, Vol. 3, No. 1, 2020.(25-37)
DOI: 10.22067/cke.2020.39287

Uncertainty-aware Path Planning using Reinforcement Learning and Deep Learning Methods*

Research Article

Nematollah Ab Azar[1] Aref Shahmansoorian[2] Mohsen Davoudi[3]

Abstract: This paper proposes new algorithms to improve Reinforcement Learning (RL) and Deep Q-Network (DQN) methods for path planning considering uncertainty in the perception of environment. The study aimed to formulate and solve the path planning optimization problem by optimizing the path, avoiding obstacles, and minimizing the related uncertainty. In this regard, a reward function is constructed based on the weighted features of the environment images. In this study, Deep Learning (DL) is used for two purposes. First, for perceiving a real environment to find the state transition matrix of the mobile robot path planning problem, and second, for extracting the features of state directly from an image of the environment to select the appropriate actions. To solve the path planning problem, it is formed in the context of an RL problem, and a Convolutional Neural Network (CNN) is used to approximate Q-values as a linear parameterized function. Implementing this approach improves the Q-learning, SARSA, and DQN algorithms as the new versions, called POQL, POSARSA, and PODQN. The learning process results show that using newly improved algorithms increases path planning performance by more than 20%, 21%, and 5% compared to the Q-learning, SARSA, and DQN, respectively.

Keywords: Reinforcement Learning, Deep Learning, Q-learning, Path Planning, Deep Q-Network (DQN).

1. Introduction

The main goal of path planning is to find the optimal path between the initial and final states in the shortest possible time[1]. This task becomes challenging when the environment has obstacles, risky areas, and uncertainties. Reinforcement Learning is a useful and applicable tool for learning path planning, considering the safety aspects of robot path, obstacle avoidance, and path optimality. Environment perception is an essential issue in path planning. The traditional learning methods use handcrafted features of the environment to recognize states-space specifications, while in the real world, the features should be used directly as the learning process input. Mnih et al. introduced the first Deep Q-network (DQN) algorithm to learn ATARI games directly from the perception of pixels [2-4]. Van Hasselt et al. [5] proposed Double DQN as a modified version of deep Q-Network. Since the introduction of deep learning, many researches have been conducted on deep learning in path planning [6-10]. Panov et al. [11] used deep learning to improve robot path planning. Pfeiffer et al. [12] presented a case study of a learning-based approach for target-driven and mapless path planning using the neural network, which is trained by a combination of expert demonstrations, imitation learning (IL), and reinforcement learning (RL). Xin et al.[7] used deep reinforcement learning in mobile robot path planning using the original visual perception with the original RGB image (image pixels) as the input without any handcrafted features and feature matching. Zhou et al. [13] implemented a Deep Q-Network (DQN) for path planning of mobile robots using the original RGB of an image representing the environmental structure. Lv et al. [14] proposed an improved learning policy DQN in a dense network framework. Concerning uncertainty-aware reinforcement learning, Kahn et al. [15] presented an algorithm for learning navigation and avoiding obstacles of a mobile robot in an unknown environment by providing an uncertainty-based cost function to estimate the probability of collision. Da Silva et al. [16] provided an action-advising framework where the agent requests for advising when its epistemic uncertainty is high in a certain state.

Using the so-called DQN concept provided by Mnih et al. [3] makes the state perception in images, as well as bulk data, learning, controlling, and making processes, actions, and decisions possible in a complex environment. Moreover, in association with RL, deep learning can approximate Q-values as a parameterized function. Using this idea increases the accuracy of Q-values by minimizing a loss function based on the gradient descent approach in the framework of neural networks containing a big hidden layer. Although reinforcement learning methods [17] such as QL and SARSA, which recently combined with DL-Q-learning called DQN [3], are reliable and stable methods for path planning, they should be improved when used for special needs such as path planning tasks. In the traditional approach of these methods, a reward function is considered as a positive integer value for reaching a goal state and is negative for others. Now, the new method proposes a new reward based on the path length, uncertainty, and constraints for every state-action pair. Thus, an optimality criterion is defined to determine how much a path is optimized and used in the path planning problem. By solving this problem in Markov Decision Process (MDP) framework, its solution is found using the learning process.

In this research, the contribution of authors is to solve the uncertain path planning optimization problem by translating it into a learning problem based on a mathematical approach, which is more complete than the works presented recently for similar purposes [14], [18- 20], and also presenting the

* Manuscript received May, 5 ,2019; accepted. November, 22 , .2020.
[1] Correspoding Author, Ph.D student, Department of Electrical Engineering, Imam Khomeini International University (IKIU), Qazvin, Iran, Email: n.abazar@edu.ikiu.ac.ir .
[2] Associate professor, Department of Electrical Engineering, Imam Khomeini International University (IKIU), Qazvin, Iran.
[3] Associate professor, Department of Electrical Engineering, Imam Khomeini International University (IKIU), Qazvin, Iran.

new methods called POQL, POSARSA, and PODQN to improve the learning process of the path planning methods considering the uncertainty in the DL results of the environment perception. In addition to those mentioned above, this article can also can be used as detailed guidance for implementing feature-based rewards and deep learning methods in path planning with visual input data [21], taking into account the practical study in a real and uncertain environment instead of the grid world approach.

The remainder of this paper is organized as follows. Section 2 discusses the related works, section 3 describes the problem statement and solution by presenting the proposed methods, algorithms, and formulations. Experimental studies are provided in section 4. Section 5 presents the simulation and results and section 6 discusses the challenges of the implementation of the proposed methods, and finally, the conclusion overview of the paper is presented in section 7.

2. Review of Related Works

This paper uses the reinforcement learning methods of Q-learning, and SARSA to implement the algorithm presented by [17], and the Q-network (DQN) algorithm developed by [3]. In these methods, state-action pairs' reward is received immediately after doing the action in the current states. The new method proposes re-evaluating the learning reward after predefined steps and building a path, including the state-action pairs of the previous steps. To describe the problem mathematically, the Lagrangian function provided by [22] is developed based on the new aspects of the problem introduced in this research to convert the path planning problem into a learning problem with the value function approach. To perceive the environment, the state transition matrix is approximated using image features by implementing the formulation presented by [23] to approximate a reward function as $R(s_t,a_t)=w^T \square (s_t,a_t)$ that is shaped by the weighting vector w and the basis function ϕ constructed as a matrix containing the logical elements representing the constraints positivity.

3. Statement of the Problem and Solution

Assume that $t\in[0\ T]$, $x\in X$ and $u\in U$ denote time, state, and control variables of the dynamic system $\dot{x}=f(x(t),u(t))$, respectively. Where T is terminal time, X, and U are the sets of feasible states and control inputs. Suppose that the variable x is measured as $\hat{x}$, and the measurement uncertainty is calculated as the mean-variance of several measurements of x as $\sigma^2(x)$. Also, it is assumed that we have some constraints. Now the path planning problem's goal is to find a trajectory $P=\{x_1, x_2, \ldots, x_n\}$, $x_1=x_0$, $x_n=x_T$ optimally, including minimizing an objective function, length, and uncertainty of the path. This optimization problem can be formulated as:

Part A of equation (1) minimizes the objective function over the states and control variables. For example, it can be considered as $l(x(t),u(t)) = (x - x_g)^T p(x - x_g) + u^T qu$. where, x_g is the robot goal position. Where q and p are the weighting matrices on state and control variables, respectively. Parts B and C minimize the functions $d_P(x)$ and $\psi_P(x)$ which are the summation of the length and uncertainty of the states of the path P, respectively. Part D is the system dynamics and $g_i(x(t))$ and $h_i(x(t))$ in part E are the constraint inequalities and equalities, respectively. Part F also represents the values of initial and terminal states.

$$
\begin{aligned}
&\min_{x,u} J(x_t,u_t)=\sum_{t=k}^{\infty}\gamma^{t-k}l(x_t,u_t) && \text{A}\\
&\min_x d_P(x)=\sum_i \|x_{i+1}-x_i\|,\ x_i\in P,\ i=1,..,n_p-1 && \text{B}\\
&\min_x \psi_P(x)=\sum_i \sigma^2(x_i),\ x_i\in P,\ i=1,..,n_p && \text{C}\\
&\text{s. t.}\quad \dot{x}=f(x(t),u(t)),\ x\in X,\ u\in U && \text{D}\\
&\qquad g_j(x(t))\le 0,\ h_i(x(t))=0,\ j=1,2,\ldots,m && \text{E}\\
&\qquad x(0)=x_0,\ x(T)=x_T && \text{F}
\end{aligned}
\qquad (1)
$$

The optimization problem (1) suffers from uncertainty, non-convexity, and non-linearity. Therefore, solving this problem is impossible or difficult using conventional and analytical methods. Thus, this paper reformulates this problem in the Markov Decision Process (MDP) context to solve it. To solve the path planning problem (1), it is assumed that the environment and the system work in a Markov Decision Process framework. In an MDP, the dynamic of the continuous system $\dot{x}=f(x(t),u(t))$ or the related discrete form $x_{t+1}=f(x_t,u_t)$ is stated as $\rho_\pi(s')=\sum_{s,a}\rho_\pi(s)\pi(s,a)T(s,a,s')$, where $\rho_\pi(s)$ and $\rho_\pi(s')$ are the state probability distribution under the policy π for the current and next states, respectively. By defining the reward function as $r(s,a)\approx -l(x_t,u_t)$ in the MDP framework, the optimization problem (1) is formed as

$$
\begin{aligned}
&\text{s. t.}\\
&\max_\pi V(s)=\sum_{s,a}\rho_\pi(s)\pi(s,a)\boldsymbol{r}(\boldsymbol{s},\boldsymbol{a})\\
&\min_x d_P=\sum_{s\in S^P}\|x(s')-x(s)\|,\ s\in S^P\\
&\min_x \psi_P=\sum_{s\in S^P}\sigma^2(x(s)),\ s\in S^P\\
&\rho_\pi(s')=\sum_{s,a}\rho_\pi(s)\pi(s,a)T(s,a,s')\\
&1=\sum_{s,a}\rho_\pi(s)\pi(s,a)\\
&\pi(s,a)\ge 0\\
&s_{t_0}=s_0,\ s_{t_f}=s_T
\end{aligned}
\qquad (2)
$$

$S^P=\{s_0,s_2,\ldots,s_T\}$ is the set of states building the Path from the initial state to the terminal state. For simplicity, let $\Pi(s,a)=\rho_\pi(s)\pi(s,a)$ and $\xi_P=d_P+\psi_P$. By developing the method proposed by [22], the Lagrangian function of the above problem is formed as

$$L=\sum_{s,a}\boldsymbol{\Pi}(\boldsymbol{s},\mathbf{a})\boldsymbol{r}(\boldsymbol{s},\boldsymbol{a})-\lambda\xi_P-\gamma\sum_{s'}V^{\pi}(s')\left(\rho_{\pi}(s')-\sum_{s,a}\Pi(s,a)T(s,a,s')\right)-r_0\left(\sum_{s,a}\Pi(s,a)-1\right) \quad (3)$$

Where $\gamma V^{\pi}(s')$ and r_0 are Lagrange multipliers, γ is the discount factor. The equation (3) can be rewritten as

$$L=\sum_{s,a}\Pi(s,a)\left(r(s,a)+\gamma\sum_{s'}V^{\pi}(s')T(s,a,s')-r_0\right)-\gamma\sum_{s'}V^{\pi}(s')\rho_{\pi}(s')+r_0-\lambda\xi_P \quad (4)$$

On the other hand,

$$\sum_{s'}V^{\pi}(s')\rho_{\pi}(s')=\sum_{s'}V^{\pi}(s')\rho_{\pi}(s')\underbrace{\sum_{a'}\pi(s',a')}_{1}=\sum_{s',a'}V^{\pi}(s')\rho_{\pi}(s')\pi(s',a')=\sum_{s',a'}V^{\pi}(s')\Pi(s',a') \quad (5)$$

Then the function L is formed as

$$L=\sum_{s,a}\Pi(s,a)\left(r(s,a)-r_0+\gamma\sum_{s'}V^{\pi}(s')T(s,a,s')\right)-\gamma\sum_{s',a'}V^{\pi}(s')\Pi(s',a')+r_0-\lambda\xi_P \quad (6)$$

Let $V^{\pi}(s_0)=0$, using the property $\sum_{s,a}V^{\pi}(s)\Pi(s,a)=\sum_{s',a'}V^{\pi}(s')\Pi(s',a')$, the function L results in

$$L=\sum_{s,a}\Pi(s,a)\left(r(s,a)-r_0+\gamma\sum_{s'}V^{\pi}(s')T(s,a,s')\right)-\gamma\sum_{s,a}V^{\pi}(s)\Pi(s,a)+r_0-\lambda\xi_P \quad (7)$$

The so-called Karush-Kuhn-Tucker (KKT) optimality conditions are implemented by differentiating w.r.t $\Pi(s,a)$ as

$$\frac{\partial L}{\partial \Pi}=r(s,a)-r_0+\gamma\sum_{s'}V^{\pi}(s')T(s,a,s')-V^{\pi}(s)=0 \quad (8)$$

Therefore, the optimal value function is obtained as $V^{*}(s)=\max_{a^{*}}\left[r(s,a)-r_0+\gamma\sum_{s'}V^{*}(s')T(s,a,s')\right]$. Letting $R(s,a)=r(s,a)-r_0$ and then the value function $V^{\pi}(s)$ results in $V^{\pi}(s)=\sum_{a}\pi(s,a)\left(R(s,a)+\gamma\sum_{s'}V^{\pi}(s')T(s,a,s')\right)$. Equivalently, the state-action value function is defined as $Q(s,a)=R(s,a)+\gamma\sum_{s'}V^{\pi}(s')T(s,a,s')$, where the transition probability function $T(s,a,s')$ should be identified. If the transition matrix is unknown, it can be obtained through environment perception. For example, in a path planning map image, a transition matrix shows how to move to the next state from the current state. One of our contributions in this paper is providing the transition matrix by extracting features of images using deep learning methods, including convolution neural networks (CNN). According to this approach, the transition matrix T is stated as $T(s,a,s')=W^{T}\emptyset(s,a,s')$, where $\emptyset$ is called the state transition feature vector, and W is the corresponding weighting vector. Also, we need to obtain r_0. By substituting $V^{\pi}(s)=r(s,a)-r_0+\gamma\sum_{s'}V^{\pi}(s')T(s,a,s')$, the term L is obtained as

$$L=\sum_{s,a}\Pi(s,a)\left(r(s,a)-r_0+\gamma\sum_{s'}V^{\pi}(s')T(s,a,s')\right)-\gamma\sum_{s,a}V^{\pi}(s)\Pi(s,a)+r_0-\lambda\xi_P$$

$$L=\sum_{s,a}\Pi(s,a)V^{\pi}(s)-\gamma\sum_{s,a}V^{\pi}(s)\Pi(s,a)+r_0-\lambda\xi_P=r_0-\lambda\xi_P.$$

Then, to minimize L, we need to consider $r_0=\lambda\xi_P=\lambda(d_P+\psi_P)$. Then, the new reward function is reached as

$$r_{new}(s,a)=r(s,a)-\lambda(d_P+\psi_P) \quad (9)$$

According to the above mathematical primary results, the new method is proposed to learn path planning based on the newly defined path optimality. This method is stated as the following algorithms:

Algorithm 1. Path Optimal Q-Learning (POQL)

Initialization: set m=number of states, n=number of actions, Q=randn (m, n), set s_s=Starting state, $\mathbf{s_g}$ =goal state, MaxIt=Maximum number of Iterations
For it=1 to MaxIt
Initialize $s=s_0$
While $s\neq s_g$
Select actions a in the state s using policy generated by Q using ϵ-greedy approach
Take the action a, get reward r, and go to state s'
Update Q-value as
$Q(s,a)=Q(s,a)+\alpha\left(r(s,a)+\gamma\max_a Q(s',a)-Q(s,a)\right)$
Store s as the path S^P and update $s\leftarrow s'$
End while
End while
For i=1 to n_p , where n_p is the length of the path P with the state set $S^P=\{s_0,s_1,...,s_{np}\}$
Update the rule
$Q(s_i,a_i)=Q(s_i,a_i)+\alpha(r(s_i,a_i)-\lambda(d_P+\psi_P)+\gamma\ \max_a Q(s_{i+1},a$
Where action a_I is connecting $s_i\in S^P$ to $s_{i+1}\in S^P$.
End For
End For

Algorithm 2. Path Optimal SARSA(POSARSA)

Initialization: set m=number of states, n=number of actions, Q=randn (m, n), set s_s=Starting state, s_g=goal state, MaxIt=Maximum number of Iterations
For it=1 to MaxIt
Initialize s= s_0
Select actions a in the state s using policy generated by Q using ϵ-greedy approach
While s≠s_g
Take the action a, get reward r, and go to state s'
Select actions $a^{'}$ in the state $s^{'}$ using policy generated by Q using the ϵ-greedy approach
Update Q-value as
$Q(s, a)=Q(s, a)+\alpha\left(r(s, a)+\gamma\, Q(s^{'}, a)-Q(s, a)\right)$
Store s as the path S^P and update $s\leftarrow s^{'}$
End while
For i=1 to n_p , where n_p is the length of the path P with the state set $S^P=\{s_0,s_1,...,s_{np}\}$
Update the rule
$Q(s_i,a_i)=Q(s_i,a_i)+\alpha\big(r(s_i,a_i)-\lambda(d_P+\psi_P)+\gamma\, Q(s_{i+1},a_{i+1})-Q$
Action a_i and a_{i+1} connect $s_i\in S^P$ to $s_{i+1}\in S^P$and $s_{i+1}\in S^P$ to $s_{i+2}\in S^P$, respectively.
End For
End For

Algorithm 1 and Algorithm 2 present the path planning instruction using modified versions of Q-learning and SARSA methods [17], respectively, and Algorithm 3 is the modified version of the DQN method [3]. The proposed path planning methods presented in the above algorithms include two levels: the first level improves the path S^P by evaluating Q-values, and the second level improves the Q-values by evaluatingpath S^P.

4. Experimental Studies

To study the theoretical method mentioned above, it should be justified and validated through practical experiments. Therefore, we examine our solution in a real environment shown in Figure 1, including a feasible path state, goal states, and Gbstacle states sets. This is a picture of the environment selected for the path planning task. In this environment, an agent can start from every feasible state to go to the Goal states (Plate Objects) by avoiding the collision states (Obstacle Objects).

Algorithm 3. Path Optimal DQN (PODQN)

Initialization: set m=number of states, n=number of actions, Q=randn (m, n), set s_s=Starting state, s_g=goal state, set replay memory D to capacity N, set Q with the initial weights θ, set MaxIt=maximum number of Iterations
For it=1 to MaxIt
Initialize s= s_0
Calculate feature ϕ(s)
While s≠s_g
Select actions a in the state s using policy generated by Q using ϵ-greedy approach
Take the action a, get reward r, and go to state s'
Calculate feature $\phi^{'}=\phi(s^{'})$
Store transition $\left(\phi(s),a,\phi(s^{'})\right)$ in D
Sample random $(\phi_j,a_j,\phi_j^{'})$ from D
Set the target value as
$y_j=\begin{cases} r_j(s, a) & \text{if } s^{'}\neq s_g \\ r_j(s, a)+\gamma\, \max_a \widehat{Q}(\phi_j, a_j^{'}; \theta) & \text{otherwise}\end{cases}$
Update the weights θ by minimizing loss function $L=\frac{1}{2}\left(y_j-Q(\phi_j,a_i;\theta)\right)^2$ using a gradient descent approach.
Store s as the path S^P and Update $Q=\widehat{Q}$, $s\leftarrow s^{'}$
End while
For i=1 to n_p , where n_p is the length of the path P with the state set $S^P=\{s_0,s_1,...,s_{np}\}$
Set the target value as
$y_i=\begin{cases} r(s_i,a_i)-\lambda(d_P+\psi_P) & \text{if } s_{i+1}\neq s_g \\ r(s_i,a_i)-\lambda(d_P+\psi_P)+\gamma\, \max_a \widehat{Q}(\phi(s_i), a;\theta) & \text{otherw}\end{cases}$
Update the weights θ by minimizing loss function $L=\frac{1}{2}\left(y_i-Q(\phi(s_i), a;\theta)\right)^2$. Where action a_i is connecting $s_i\in S^P$ to $s_{i+1}\in S^P$
Update $Q=\widehat{Q}$
End For
End For

Figure 1. An experimental environment for testing the new method(Path Optimal RL(PORL) and Path Optimal DQN(PODQN))

To learn the path planning task, the following procedures are proposed:

4.1. Convert real environment to a grid world

In most cases, a real environment must be converted to the discrete-time domain because the dynamic system or the problem's solution is presented in a discretized form. For this purpose, the environment is mapped to a two- or three-dimensional grid network (x-y or x-y-z cartesian space). If the problem's variables are fixed, or their changes are negligible in the third coordinate (that is, z), it is better to consider the environment as a two-dimensional space. In this research, the picture of environment is analyzed in a 2-d grid world space with n×m nodes. Where n and m are the length and width of the picture, respectively. In this framework, the states represent the agent positions. Moreover, the actions are the velocity and angular velocity of the robot and can be stated as the movement of the agent form the current state (cell i and j) to the next state (cell i' and j') (with $\Delta i, \Delta j=\pm 1, \pm\sqrt{2}$) in a unique grid world. Then, the actions can be described as moving to the right, left, up, and down. So, for the dynamic environment, the sets of states and actions are presented as $S=\{s_1,s_2\ldots,s_{m\times n}\}$ and A={right, right-top, top, left-top, left, left-down, down, right-down} where, s_k is the agent state. Figure 2 shows the grid world environment and an example of the action set.

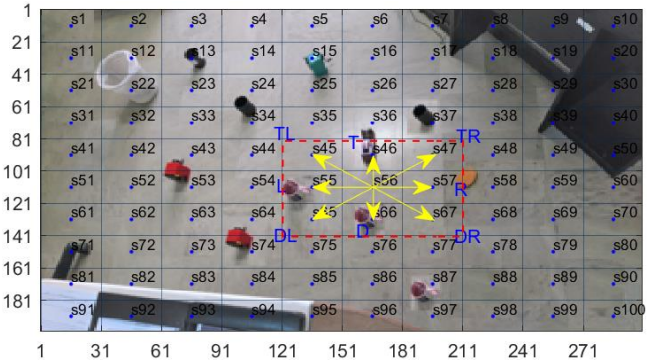

Figure 2. Path planning environment, including feasible, goal, and collision states. The arrows show the actions set 1. R: right, 2. TR: top-right, 3. T: top, 4. TL: top-left, 5. L: left, 6. DL: down-left, 7. D: down, 8. DR: down-right.

As Figure 2 shows, an agent can go to the next state from the current state by taking appropriate action. In this study, the next states are neighbors of the current state with one neighborhood radius unit. In Q-learning, SARSA, and DQN methods, the next action is selected by evaluating a probability distribution of the actions set connecting the current state to the next states. After selecting the action, a reward is also assigned to the corresponding state-action pair, and a discounted Q-value accumulates this reward. This Q-value used to build the probability distribution mentioned above must be predicted and updated for the new step-time. A learning process is satisfactory when the cumulative discounted value, called target value from now on, approaches to the predicted Q-value of the current state-action pair. In other words, the error between the target and the predicted values of Q must be decreased step by step. The average accumulated rewards obtained in the learning time must also converge to the two values in every time step.

4.2. Identification of state transition matrix

A state transition matrix, often used in the Markov decision process, is a stochastic or probability matrix representing the transition between two states (describing how to go to the next state from the current state). For example, the corresponding transition matrix of the system $x_{k+1} = f(x_k,u_k)$ is stated as $T(s, a, s')=p(s_t, a_t|s_{t+1}=s')$. In the visual scheme, as shown in the red box in Figure 2, it can be obtained for the state s_{56} by taking actions from the arranged set (R, TR, T, TL, L, DL, D, DR) as $T(s_{56},:,:)=\left[\frac{1}{5}\ \ \frac{1}{5}\ \ 0\ \ \frac{1}{5}\ \ 0\ \ \frac{1}{5}\ \ 0\ \ \frac{1}{5}\right]$ where the denominator of 5 represents the number of feasible states that do not contain collisions. A pure probability matrix can also represent it as $T(s_{56},:,:)=[\xi_1\ \ \xi_2\ \ \cdots\ \ \xi_8]$ where ξ is the probability measure of the state feasibility. To find the transition matrix, one can employ the handcrafted feature extraction and states' classification (into three classes: feasible, goal, and obstacle sets or the number of all objects in the image) using deep learning methods.

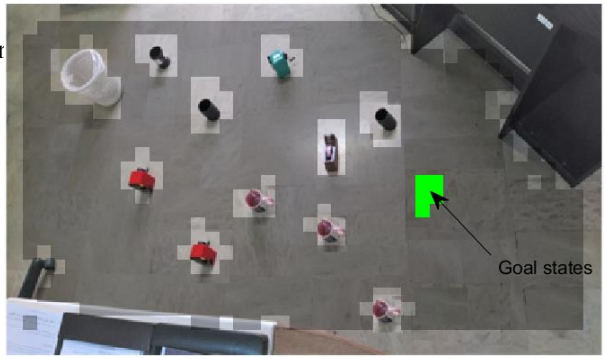

Figure 3. The environment, including feasible states, obstacle states, and goal states, are identified.

When the environment is identified, as shown in Figure 3, the transition matrix can be built. Knowing the transition matrix, a reward matrix is designed by assigning +5, -5, and -1 to the actions leading to the goal, obstacle, and feasible states from the feasible states.

4.3. Calculating uncertainty

As mentioned in the previous section, to perceive the environment, objects in the environment are classified using the Deep Learning and Convolutional Neural Network (CNN) concept, and the probability that an object is placed in a class is calculated using a vector of SoftMax values. Here, uncertainty means inaccuracy in identifying objects in the environment using classification theory. For example, if the probability of a pixel belonging to the obstalce class is obtained equalt to 0.95, then the uncertainty will be 0.05. That is, we will have 0.05% uncertainty in identifying the class related to this pixel by deep learning. Figure 5 shows the uncertainty measures for all the environment states related to Figure 3.

4.4. Path Planning using RL

After identifying the transition matrix and assigning rewards to every state-action pair, path planning learning starts based on the related algorithms of RL; Q-learning, and SARSA, which are referred to by [17]. Figure 5 shows the schematic of the interconnection between the reinforcement learning methods (QL and SARSA), system, and the value function.

4.5. Path Planning using DQN

To implement path planning in a DQN framework, the Q-values must be predicted using image pixels as a Convolutional Neural Network input. In this context, the images of the current state and the next states (the neighbors of current states) are used as the input of the convolution layers to provide a feature vector as the additional neural network input. The proposed CNN is shown in the following:

Suppose selecting any action from the actions set $a=a_1,\ldots,a_8$ in the current state s_t leads to the corresponding next states $s'=\{s'_1, s'_2, \ldots, s'_8\}$, a feature vector can be defined as the feature of image pixels outlined in the states s. As shown in Figure 6, the image pixels of the states of the environment are the input of the convolution layer, and its output is a feature vector for every state. To better understand, Figure 7 presents an example of a feature vector arrangement for the current state s_{56} and next states $s'=\{s_{57}, s_{47}, s_{46}, s_{45}, s_{55}, s_{65}, s_{66}, s_{67}\}$. Since every action from the current state leads to a different state, an action can be predicted by knowing the current and next states. For this purpose, the Q-value is approximated by the linear combination of features of the current and next states as

$Q(s, a)=W_1 F(s)+W_2 F(s')+b=[W_1 \; W_2]\begin{bmatrix}F(s)\\F(s')\end{bmatrix}+b=W\phi(s, a)+b$. Where, $\phi(s, a)=\begin{bmatrix}F(s)\\F(s')\end{bmatrix}$ is called feature vector of state-action pair (s, a). After providing the feature vectors, the DQN algorithm is implemented by referring to [3]. The schematic of the interconnection between the DQN, system, and environment is shown in Figure 8.

.

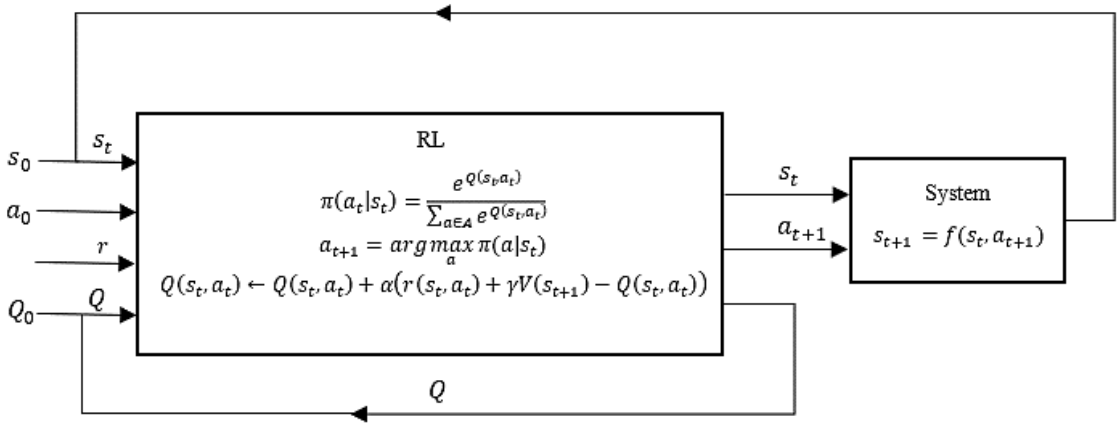

Figure 4. Interconnection between the reinforcement learning(QL), system, and value function

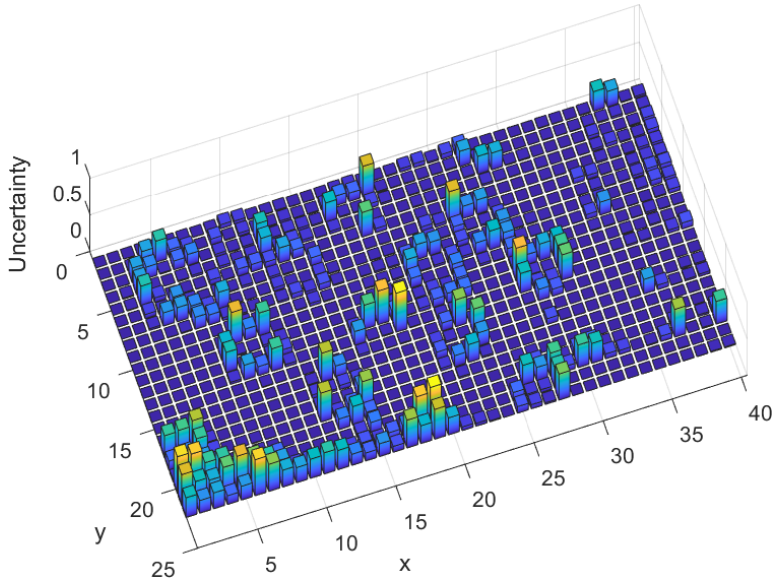

Figure 5. Uncertainty measures for all the states of the environment

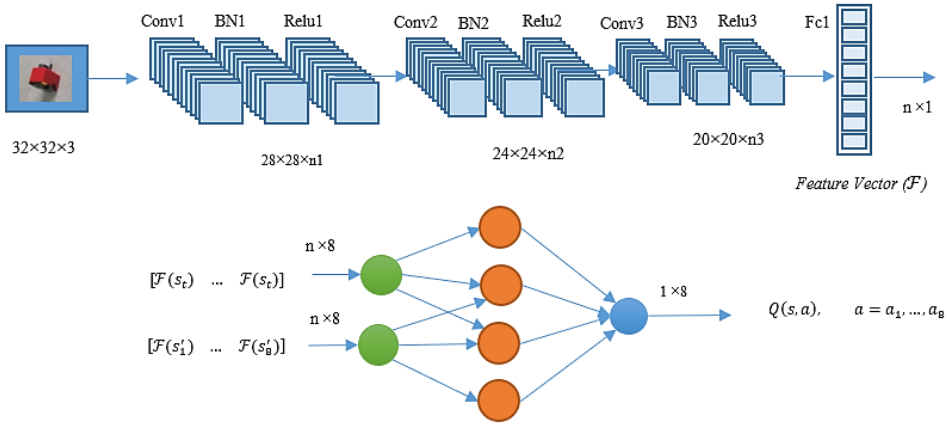

Figure 6. The proposed Convolutional Neural Network (CNN)

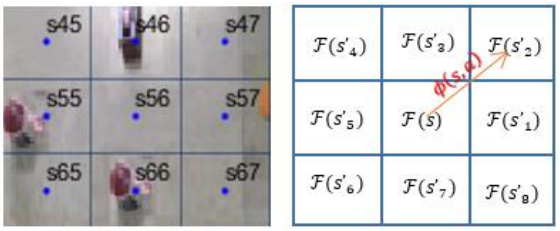

Figure 7. Feature vector arrangement for current state s and next states s'

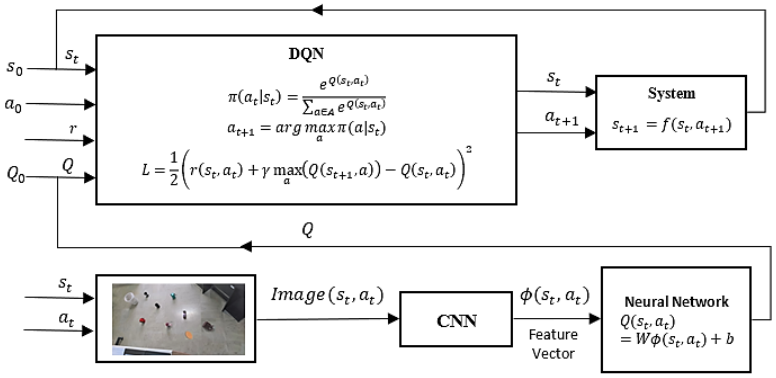

Figure 8. The reinforcement learning (RL), system, value function, and deep learning (DL) in the proposed method

5. Simulation and results

In the first phase, the mentioned learning methods (Q-learning, SARSA, and DQN) were implemented on the path planning task in the environment simulated in MATLAB. The program is run in 10000 Episodes(iterations) and 200 maximum step (MaxStep) with a learning rate of α=0.5 for Q-learning and SARSA, and $\eta=\frac{\eta_0}{1+\frac{\text{Episode}}{\text{MaxEpisodes}}}$, η_0=0.02 for DQN, a discount factor of γ=0.9, and ε =0.3 for the ε -greedy approach. In the second phase, the new proposed method is implemented in the mentioned learning method. The Convolutional Neural Network (CNN), described in section 4.5, is used with 7 layers including 3 Convolution-Batch Normalization-Relu layers arranged with each other, 1 fully connected layer for creating the feature vector as Conv1-BN1-Relu1--Conv2-BN2-Relu2--Conv3-BN3-Relu3--fc4, and 2 fully connected and 2 Relu layers for building the neural network as fc5-Relu5--fc6-Relu6--fc7. Figure 9 shows the averages of rewards, steps, time steps, and errors obtained from implementing the old and newly proposed methods. Excluding the average of time steps, the performance of new method has improved in terms of other factors. By implementing the newly proposed method, the number of steps per episode, and the learning error have decreased, and the average reward has increased. These are the advantages of the new methods, but the run time of learning in the new methods is longer than the old methods, which is a disadvantage of our new methods.

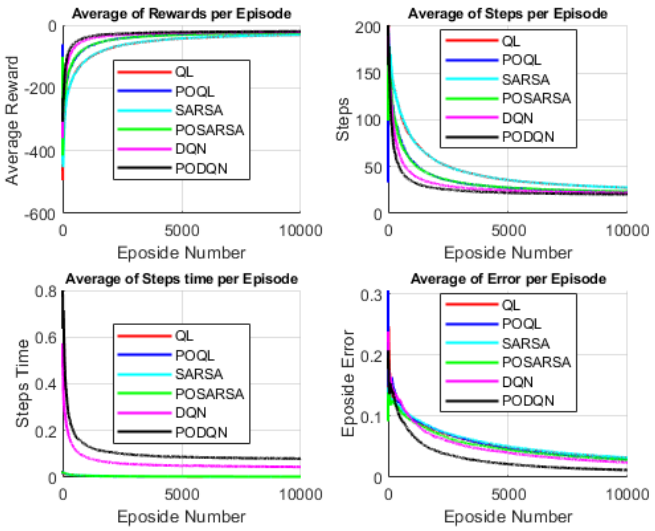

Figure 9. Comparison of learning methods on Averages of Rewards, Steps, Steps time, and Error.

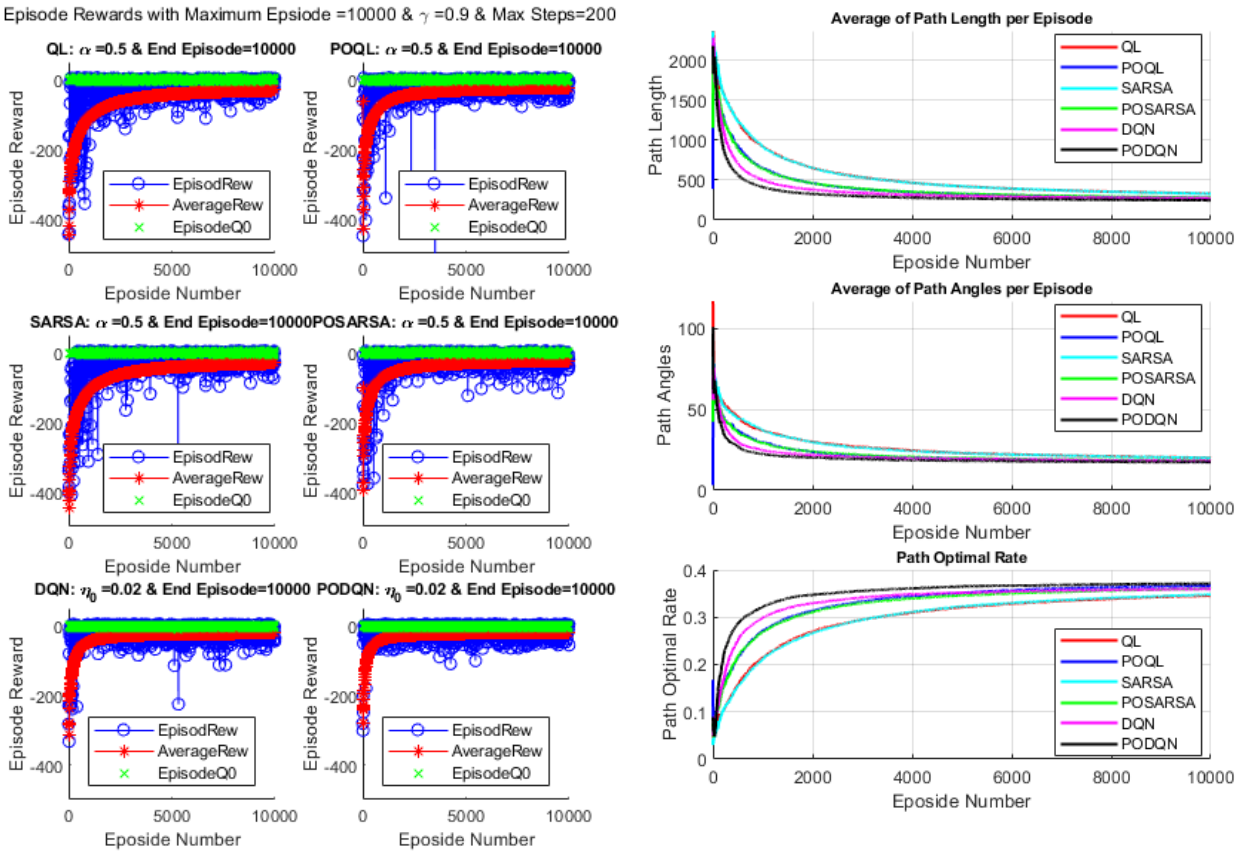

Figure 10. Left: the everage of rewards, target, and predicted Q-values for the new methods (i.e., POQL, POSARSA, PODQN) and the old methods (i.e., QL, SARSA, DQN). Right: Averages of Path length, angles and optimal rates for both new and old methods.

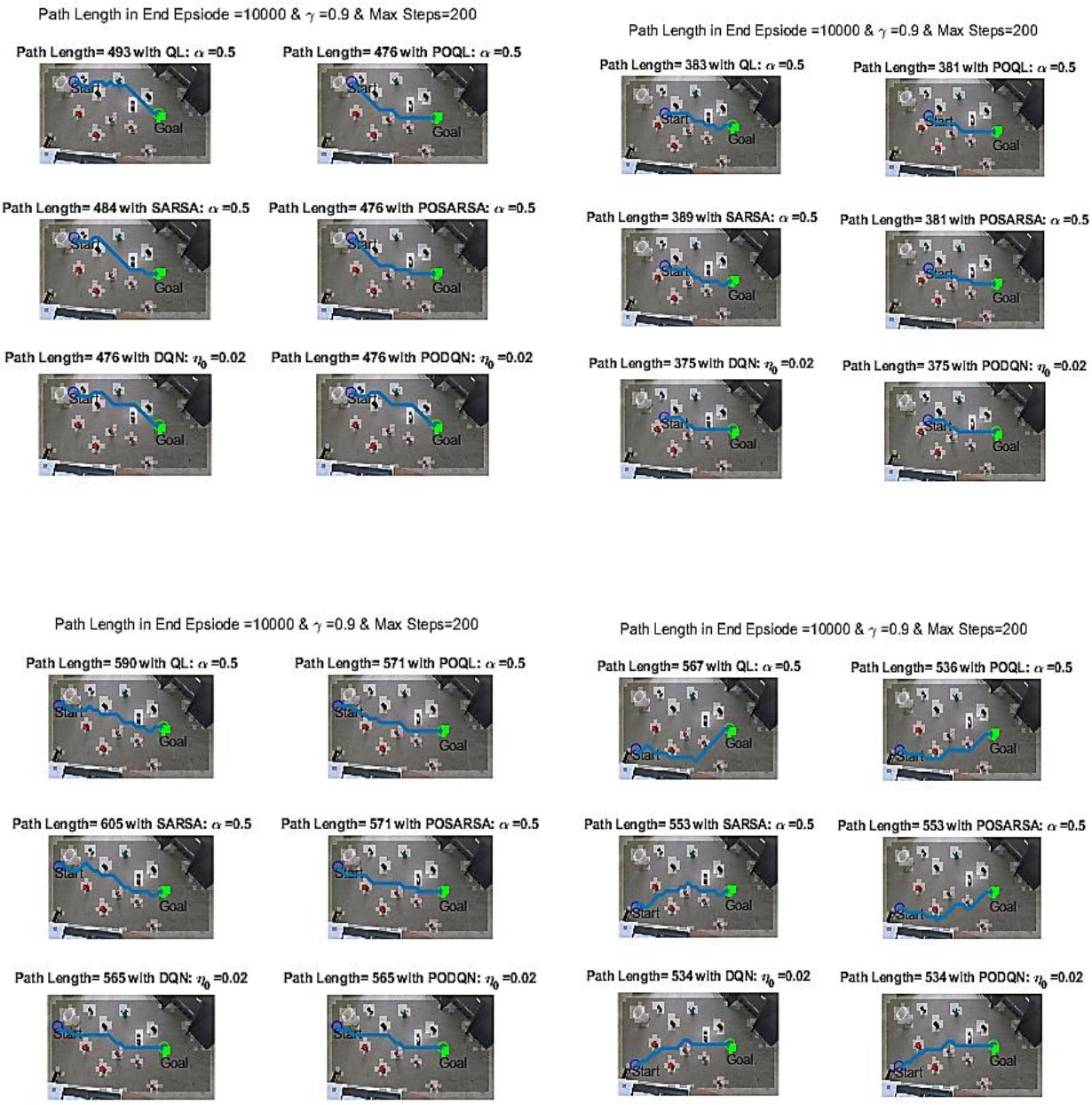

Figure 11. Compairison the proposed method with new method for path plaing by starting from several scenarios

In Figure 10, the image on the left shows that the average rewards, target, and predicted Q values converge in descending order, resulting in a steadily decreasing learning error. The new methods (POQL, POSARSA, PODQN) have received fewer negative rewards rather than the old methods (QL, SARSA, DQN). In other words, the negative zone is narrower than for the new methods compared to old methods. Moreover, in the right image, the averages of path length and the averages of the sum of path angles for the new methods are less than that of the old methods.

Figure 11 shows the test results of four initial states (start points) s_{128},s_{331},s_{242},s_{644}, and the predefined goal states (plate shape remarked by green color). As shown in all tests, the path length obtained by the new methods (POQL, POSARSA, and PODQN) is shorter or equal to that of the old methods (QL, SARSA, and DQN). It means that the path optimality of the new method is more than the old methods.

Figure 12 shows the average of the sum of uncertainty measures for all the states of every path in the iterations. The results indicate that the uncertainty is gradually decreasing by increasing the iterations for all methods. Besides, the newly proposed method minimizes the path uncertainty much better than the old methods.

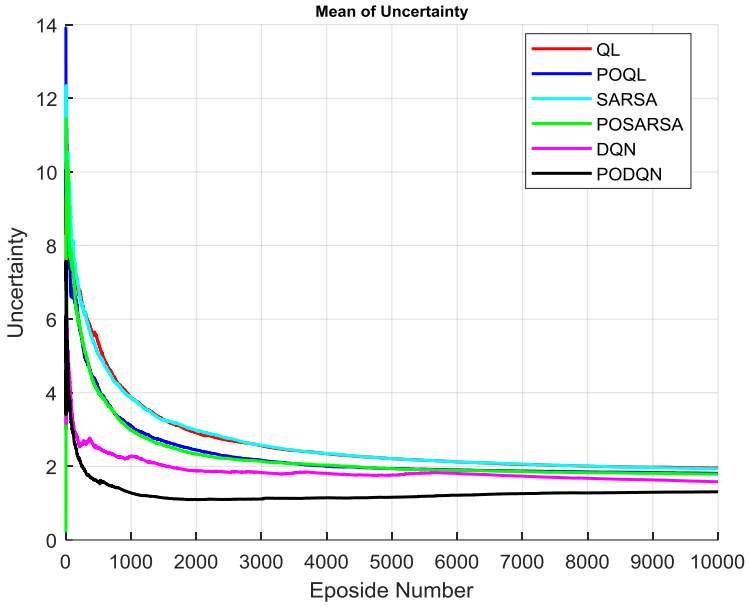

Figure 12. The average of the sum of uncertainties for every path obtained in the iterations

6. Discussion

While implementing RL and DQN and the new method, one may encounter some challenges, including data availability, dimensionality, visual perception by feature extraction, learning stability, and butterfly effects. Regarding data availability, learning problems need massive datasets for training. Fortunately, the required data can be provided for the path planning problem. One can obtain this data from the real environment and events in our daily life, from the personal houses to roads and highways, and by taking pictures and videos. Dimensionality is the most critical challenge because, as mentioned above, learning problems are valid by using massive datasets. However, to solve this problem, one needs to use a well-equipped computer with high CPU and GPU performance. Although we need to run the program for a few hours or days in many cases of deep learning, in this study, we apply some assumptions and provisions to reduce dimensionality. For example, by selecting images just from states required for evaluation in a specific state, the computations are decreased. Moreover, using toolboxes may increase the run time. So, one can code the required algorithm by himself to remove the irrelevant scripts and functions.

Visual perception and feature extraction are other issues in deep learning. Selecting appropriate features plays a crucial role in the perception of the environment. In the this study, we needed to approximate Q-values using image features. Finding the proper features requires examining the inputs by changing CNN, layers, parameters, and hyperparameters. Moreover, overfitting data is hazardous when the number of parameters dramatically exceeds the number of independent observations. One can use regularization, normalization, and limitation on learning weights and required parameters to avoid such problems. Finally, the learning process is strongly sensitive to the butterfly effects as small variations in the input data might lead to extremely different results and thus learning instability. For example, selecting image features and also the learning rate is essential for deep learning. As a piece of good news, learning results are achieved by averaging the outputs in a long time or significant steps, and it can reduce the effect of small local disturbances in the whole process. However, for stochastic gradient descent algorithms such as DQN, the inputs and parameters must be selected reliably. In this way, it might be needed to monitor outputs changes after selecting or changing the inputs to find the best parameters by trial and error if necessary.

7. Conclusion and future works

In this paper, a new method was proposed to improve the learning algorithms, including Q-learning, SARSA, and DQN, for path planning tasks to minimize the path length and uncertainty. The results illustrate that the new method can improve path planning optimality at least 21%, 21.5 %, and 5-8% for POQL, POSARSA, and PODQN compared to QL, SARSA, and DQN, respectively. Moreover, the challenges of applying the new method were discussed. In future studies, this method can be applied to other benchmarks in reinforcement learning and deep learning methods and problems such as highway and traffic issues.

References

[1] M.Samadi and M. F. Othman, “Global path planning for autonomous mobile robot using genetic algorithm”, *In Signal-Image Technology & Internet-Based Systems (SITIS), 2013 International Conference on* “, pp. 726-730, IEEE, 2013.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing

atari with deep reinforcement learning", *arXiv preprint arXiv:1312.5602*, 2013.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, andS. Petersen, "Human-level control through deep reinforcement learning", *Nature*, 518(7540), 529, 2015.

[4] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning", *In International Conference on Machine Learning*, pp. 1928-1937, 2016.

[5] H. Van Hasselt, A. Guez, A., and D. Silver, "Deep reinforcement learning with double q-learning", *In Thirtieth AAAI Conference on Artificial Intelligence*, 22016.

[6] M. Wulfmeier, D. Rao, D. Z. Wang, P. Ondruska, and I. Posner, I. "Large-scale cost function learning for path planning using deep inverse reinforcement learning", *The International Journal of Robotics Research*, 3936(10), 1073-1087, 2017.

[7] J. Xin, H. Zhao, D. Liu, and M. Li, "Application of deep reinforcement learning in mobile robot path planning", *In 2017 Chinese Automation Congress (CAC)*, pp. 7112-7116, IEEE, 2017.

[8] Y. F. Chen, M. Everett, M. Liu, and J.P. How, "Socially aware motion planning with deep reinforcement learning", *arXiv preprint arXiv:1703.08862*, 2017.

[9] U. Challita, W. Saad, and C. Bettstetter, "Deep reinforcement learning for interference-aware path planning of cellular-connected UAVs", *In Proc. of International Conference on Communications (ICC)*, Kansas 20 City, MO, USA, 2018.

[10] Y.H. Kim, J. I. Jang, and S. Yun," End-to-end deep learning for autonomous navigation of mobile robot", *In Consumer Electronics (ICCE), 2018 IEEE International Conference on*, pp. 1-6, IEEE, 2018.

[11] A. I. Panov, K. S. Yakovlev, R. Suvorov, "Grid path planning with deep reinforcement learning: Preliminary results", *Procedia computer science*, 123, 347-353. 2018.

[12] M. Pfeiffer, S. Shukla, M. Turchetta, C. Cadena, A. Krause, R. Siegwart, J. Nieto, "Reinforced Imitation: Sample Efficient Deep Reinforcement Learning for Mapless Navigation by Leveraging Prior Demonstrations", *IEEE Robotics and Automation Letters*, 3(4), 4423-4430, 2018.

[13] S. Zhou, X. Liu, Y. Xu, J. Guo, "A Deep Q-network (DQN) Based Path Planning Method for Mobile Robots", *In 2018 IEEE International Conference on Information and Automation (ICIA)*, pp. 366-371, IEEE, 2018.

[14] L. Lv, S. Zhang, D. Ding, Y. Wang, "Path planning via an improved DQN-based learning policy", *IEEE Access*, 7, 67319-67330, 2019.

[15] G. Kahn, A. Villaflor, V. Pong, P. Abbeel, S. Levine, "Uncertainty-aware reinforcement learning for collision avoidance", *arXiv preprint arXiv:1702.01182*, 2017.

[16] F. L. Da Silva, P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "Uncertainty-Aware Action Advising for Deep Reinforcement Learning Agents", *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(04), 5792-5799, 2020.

[17] R. S. Sutton, and A.G. Barto, "Introduction *to reinforcement learning*, Vol. 135, Cambridge: MIT press, 1998.

[18] M. W. Otte, "A survey of machine learning approaches to robotic path-planning", *University of Colorado at Boulder*, Boulder, 2015.

[19] X. Lei, Z. Zhang, and P. Dong, "Dynamic path planning of unknown environment based on deep reinforcement learning", *Journal of Robotics*, 2018.

[20] T. Blum, W. Jones, and K. Yoshida, "Deep Learned Path Planning via Randomized Reward-Linked-Goals and Potential Space Applications", *arXiv preprint arXiv:1909.06034*, 2019.

[21] S. Lange, M. Riedmiller, and A. Voigtländer, "Autonomous reinforcement learning on raw visual input data in a real world application", In *The 2012 international joint conference on neural networks (IJCNN)*, pp. 1-8, IEEE, 2012.

[22] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey", *The International Journal of Robotics Research*, 32(11), 1238-1274, 2013.

[23] P. Abbeel, and A.Y. Ng, "Apprenticeship learning via inverse reinforcement learning", *In Proceedings of the twenty-first international conference on Machine learning*, p. 1, 2004.