JAES | JOURNAL OF APPLIED ENGINEERING SCIENCE | www.engineeringscience.rs
ISTRAŽIVANJA I PROJEKTOVANJA ZA PRIVREDU

Indexed by

# TRAINING NEURAL NETWORKS USING REINFORCEMENT LEARNING TO REACTIVE PATH PLANNING

***Milton Vicente Calderón***
*Universidad Distrital Francisco José de Caldas, Faculty of Technology, Department of Electronic, Bogotá D.C, Colombia*

***Esperanza Camargo Casallas***
*Universidad Distrital Francisco José de Caldas, Faculty of Technology, Department of Electronic, Bogotá D.C, Colombia*

***Key words***: *path planning, reinforcement learning, neural network*

**Cite article**:
Milton, V. C., & Esperanza, C. C. [2021]. Training neural networks using reinforcement learning to reactive path planning. *Journal of Applied Engineering Science*, 19(1) 48 - 56. DOI:10.5937/jaes0-26386

***Online aceess*** *of full paper is available at: www.engineeringscience.rs/browse-issues*

Istraživanja i projektovanja za privredu
Journal of Applied Engineering Science

Original Scientific Paper

**doi:**10.5937/jaes0-26386

Paper number: 19(2021)1, 762, 48-56

# *TRAINING NEURAL NETWORKS USING REINFORCEMENT LEARNING TO REACTIVE PATH PLANNING*

***Milton Vicente Calderón*, Esperanza Camargo Casallas***

***Universidad Distrital Francisco José de Caldas, Faculty of Technology, Department of Electronic, Bogotá D.C, Colombia***

*The mobile robots are devices with great boom given the possibilities that their utilities offer, and to a greater extent, those freelancers who do not require an operator to perform their functions. In order to consolidate the autonomy it is necessary to generate a system of planning of ways that allows a viable route and as far as possible optimal. This study develops a reactive two-dimensional path planning method with neural networks trained under the reinforcement learning method. The complexity of the scenario between the initial and final point is due to warning and forbidden obstacle zones, and the experimentation is carried out on different neural network architectures, each one as an agent of the learning-by-reinforcement algorithm, being these DQN and DDQN types. The best results are obtained with the DDQN training, reaching the objective in 89% in the validation episodes, although the DQN method shows to be 15.63% faster in its success cases. This work was carried out within the research group DIGITI of the Universidad Distrital Francisco José de Caldas.*

*Key words: path planning, reinforcement learning, neural network*

## *INTRODUCTION*

Technological development has enabled agents to carry out protection, identification/recognition, tracking, automation of processes and weapons activities [1]. This development has led to the creation of robots with specific architectures that are capable of extracting and interpreting data according to diverse functionalities [2, 3], including the ability to navigate autonomously [1, 4], and plan globally or locally [5], whether this is known or uncertain. There is a method called ”reactive”, which happens when planning is local and experiences an uncertain environment [2, 3, 6].

The central objective of mobile robotics is to generate controlled displacement from an initial/current point to an end point, so that the route traced ensures the survival of the robot and is as efficient as possible, however, such survival involves avoiding obstacles that may affect the integrity of the robot [7, 8]. For this reason, path planning is vital for the implementation of intelligent robots, since the appearance of unknown and complex environments must be taken in account[9, 10]. Thus path planning becomes an optimization problem in which solution lies in finding a valid and optimized path, avoiding obstacles and strengthening safety [2].

With these developments, not only are reliable route methodologies established for missionary robots, but also a potential is extended with application to travel vehicles in urban areas, [6], resulting in reduced times and safety as a measure against accidents.

For path planning, the use of algorithms oriented to cartographic processes has been required, being relevant those that use deep neural networks [9], genetic algorithms [2], geometric optimization algorithms [1], bio-inspired algorithms in bacterial survival [10], heuristic [3], neuro-diffusion [4], Qlearning [11] and gradient and lagrangian optimization methods [6], all oriented to make the mobile robot reach the target efficiently in different environments and with various obstacles in several ways.

Under the circumstances exposed in this paper, it is determined to experiment a reactive path planning model, that is, one that reacts to the environment state, using neuronal networks with learning by reinforcement, studying its DQN and Double DQN variants, to conform a system that responds through an agent to a state representation that preserves the environment observations. The whole of this paper is organized as follows. First is the framework, where there is a brief explanation of the path planning, the reinforcement learning algorithms and background work to this article. Later, we find the methodology, which includes the neuronal training algorithm used and the process of experimentation carried out. Then, the results obtained from the generated systems are shown and finally there is the conclusion.

## *FRAMEWORK*

### *Path Planning*

The trajectory generation is a crucial issue in the robotic field with application in scientific and industrial areas [5], fundamental for autonomous mobile robots, which have the challenge of interacting with the environment [4], generating the need to predefine routes between a starting point and an end point [2, 5, 11]. The elementary situation is when the generation is done on a known and static environment, such generation can be called deliberate. However, the generation of paths can be established for an unknown and changing environment which is called reactive because it reacts to the state of the

**mvcalderonc@correo.udistrital.edu.co*

environment [1, 3, 6, 12].

The path planning systems can also be classified as local or global [2], being local the one that uses real time information to execute actions in a dynamic way, and global the one whose path is totally defined depending on the start/end points considering static obstacles [1, 2, 3, 5, 9, 10], therefore, due to the commitment with a reactive model, with dynamics in the environment and local exploration, special emphasis should be placed in generating a path that can be performed at high speed and protecting the robot from any damage either by a collision with an obstacle or by excessive accelerations or vibrations.

The algorithms used for the creation of trajectories in a local way have the necessity of constructing the map of the environment [10], whose variants are grouped according to the methodology used to generate the trajectory [13, 14]:

- Roadmap techniques.
- Cell decomposition.
- Artificial potential.

Analytical, enumerative [10] or computational intelligence methods also develop these variants[1, 2, 3, 4, 5, 6, 9, 10, 11]. Despite the fact that the main problem arises with the complexity of search methods for navigating the unknown, a reinforced learning based on a set of rewards and punishments depending on free or truncated movement due to collisions addresses the matter[11].

### *Reinforcement learning*

Reinforcement learning, unlike other artificial intelligence algorithms, is a learning method that does not require any rules [15], and is based on relating a situation or state to an action in order to maximize the reward [16]. The learning agent is not instructed on what action to take, instead he must explore the available actions and determine the reward obtained by performing one or another action [11]. The agent must be able to obtain an observation of the state of his or her environment and take an action that generates an effect on the state of the environment, having a goal or goals related to the state of the environment. Figure 1 shows the interaction of the different factors involved in the algorithm.

### *QLearning*

QLearning is a relevant type of reinforcement learning thanks to the conjunction of the principle of penalty and reward with the interaction of the robot with the environment [11]. The algorithm focuses on value-based reinforcement learning [17] that is updated as the environment is explored by means of the Q-value function. This type of storage presents a big problem when the number of possible states is very high because it makes the table to have many registers, this generates that the search is slower, and also, a high number of states can be caused because a dimension is added for each input/output variable, or when the state variables are continuous or have very small variations [11, 18, 19]. The function that governs Q-value is,

$$Q_{target}(s,a) = r + \gamma maxQ(s^0,a^0) \quad (1)$$

Where $r$ is the reward received, $\gamma$ E [0,1] is the discount factor that seeks to maximize future rewards over the immediate and $maxQ(s^0, a^0)$ is optimal value for the state $s^0$ and the action $a^0$.

### *Deep Q-Network (DQN)*

For such a system, the correspondence between state/observation, reward, and action, is learned by a neural network through the approximation function [19], which represents an improvement over QLearning since no search is performed, but an evaluation of the neural network. However, to perform the neural network training, storage space must be created for the correspondences obtained during the training phase. In this case the following expression is used,

$$Q_{target}(s,a) = r + \gamma maxQ(s0,a0;\theta_{target}) \quad (2)$$

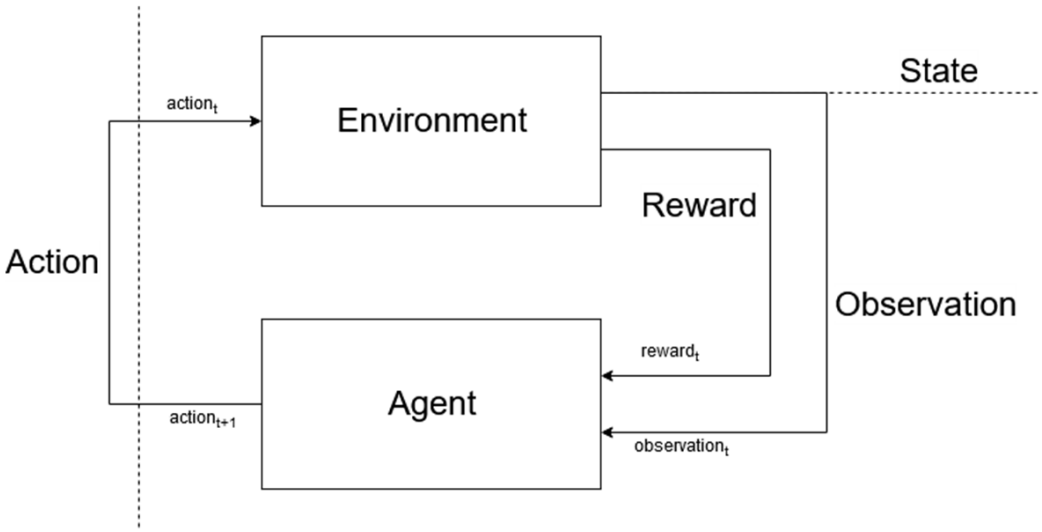


*Figure 1: Reinforcement learning diagram*

Where $\theta_{target}$ represents the parameters that will be modified during the training process.

### *Double DQN*

As DQN the correspondence process between state, reward and action is done through a neural network, however it has been established that it presents an overestimation problem due to the approximation function is insufficiently flexible [20] or by noise [21, 22], so that is why it is necessary to reinforce the learning with a second neural network which aims to minimize the overestimation, thus generating the following mathematical model,

$$Q_{target}(s,a) = r + \gamma Q(s0,a0 = argmaxQ(s0,a0,\theta_{eval});\theta_{target}) \quad (3)$$

For which $\theta_{eval}$ represents the parameters of the neural network used to control the overestimation problem described above.

### *Related Works*

The search for effective methodologies for path planning has generated a large amount of work. A general analysis is made by [3], which studies classical and heuristic methods. Among their results, it is found that even though classical methods are easier to implement, they are not so ”intelligent” since, unlike heuristic methods (neural networks, diffuse, bio-inspired and hybrid algorithms), they do not have the capacity to adapt and generalize, however, they have the disadvantage of requiring a higher computational cost. Finally, hybrid systems with neural networks, such as those presented in [4] to generate a recognition system through artificial vision with a genetic algorithm validated in practice, and [2], where genetic optimization algorithms are used to improve the efficiency of the path, reducing the number of iterations required to establish a path on a 2D plane of static environment.

Also, among those models based on optimization algorithms, bioinspired or purely mathematical is [1], which discusses a geometric method called Coordinate Reference Framework for a static environment formed by elliptical obstacles with effective results in a 2D environment. In the same way [10] explores an optimization algorithm based on bacterial foraging (BFO) with a Gaussian cost function, demonstrating better results compared to swarming and basic BFO by establishing a path due to particles distributed around the robot, a fact related to the method exposed in [5], which addresses the problem of trajectory planning based on the theory of potential fields of charged particles, assigning a potential function to each obstacle that interacts with the scalar surface depending on the characteristics of the robot, and obtaining a good balance between fast approach to the target and number of collisions, a peculiarity that is shared with the results of [6], which certifies the functionality of optimization algorithms through the gradient and lagrangian method with the Biezer curve, but demonstrates that in the path fitting process considerable distortion is caused with respect to the initial path as a response to avoid obstacles.

Regarding the use of reinforcement learning algorithms, studies such as [9, 11, 15, 17] show the need to optimize the processing time. The study [15] recognizes its slow convergence, although it shows that with Qlearning and the Nomoto equation after sufficient rounds of training the system is effective in self-learning and continuous optimization. On the other hand, [11] implements as a partial guide the flower pollination algorithm, generating a good basis for initialization that accelerates the learning process and that wins over basic Qlearning in terms of path optimization and computation time reduction (reduction from 10 to 13%), also establishing the potential integration of a neural network for state processing, fact realized by [17], that finds advantages in terms of processing speed, and that corresponds to the study [9], which generates the construction of a 3D trajectory planning system with Qlearning and a deep neural network in order to improve the processing time, as well as to achieve low computational disturbance in the face of the variability of different complex environments.

## *METHODOLOGY*

### *Problem*

The generation of a path planning system for a mobile robot in a 2D environment is determined. Given the background, the use of learning by reinforcement algorithms is defined to help a feed forward neural network in order to generate DQN and DDQN experimentation methodologies. For this it is necessary to establish the development environment, with its predefined dynamic and static characteristics, as well as the representation of states, the reward instances, and the agent properties.

### *Environment*

The environment consists of a two-dimensional space in which the following elements are established:

- Environment coordinates (given by four coordinates).
- Object step length.
- Available actions.
- Set of measure points fixed in relation to vehicle location
- Start point.
- End point.
- Obstacles or zones:
- Danger zones, which should be avoided but can be crossed.
- Prohibited areas, which would involve the destruction of the vehicle.

Of the elements or properties described above, the environment coordinates, object step length, available ac-

tions and measurement points are maintained throughout the experiment, while the remaining properties change randomly as soon as it is established that the object reached one of the following conditions:

- Reaches the destination point.
- Entered one of the forbidden zones.
- It left the limits of the environment.
- The maximum number of steps per episode was reached.

### *State representation*

The state representation corresponds to a set of environmental data available to the agent, such data set could be established as an observation of the immediate environment state through which the agent determines an action to be performed on the environment with which it will modify its current state.

For the experiment the state representation contains the following information:

- Unitary vector with origin in the vehicle and pointing to the destination.
- The current orientation angle of the object.
- Measurement points located in relation to the object.

### *Reward*

The reward is a value by which the environment indicates to the agent the potential of the current state, to determine this value the following conditions are taken into account:

- Test points outside the environment
- Test points contained in prohibited areas
- Test points contained in danger zones
- The vehicle is located outside the environment
- The vehicle is located in a prohibited area
- The vehicle is located in a danger zone
- The vehicle reaches the destination point
- None of the above

### *Agent*

The agent is composed of a neural network which is in charge of establishing the relationships between the observations of the environment, the actions performed and the rewards obtained, using a storage structure for the data tuples used throughout the training iterations with which the neural network training will be performed. Additionally, it must be able to execute three actions:

- Act: determines an action and interacts with the environment to modify its status.
- Save: stores a tuple of state, action and reward data.
- Learn: performs neural network training with previously stored data.

### *Training process*

### *Algorithm*

In order to carry out the training process of neural network by mean of reinforcement learning an algorithm was established with two modes(train and test) and its main activities diagram is shown in figure 2, in that figure, can be observed the following steps: firstly, it initializes the environment and an agent, these variables will be kept constant during the experiment, in test mode the agent has to load additionally a saved configuration from a previous training experiment; secondly, it initializes episode variables, those variables will be used just in one episode, in both modes works equal; thirdly, it starts the interaction between environment and agent, in both modes the agent gets the environment status and then push an action to environment changing its status, in training mode the action chosen at firsts steps is completely random getting an environment exploration, while the learning process go forward the agent in order to determine a better action using the environment status.

The data required to train the agent and then perform its learning process is generated by interaction between environment and agent, that interaction is executed until

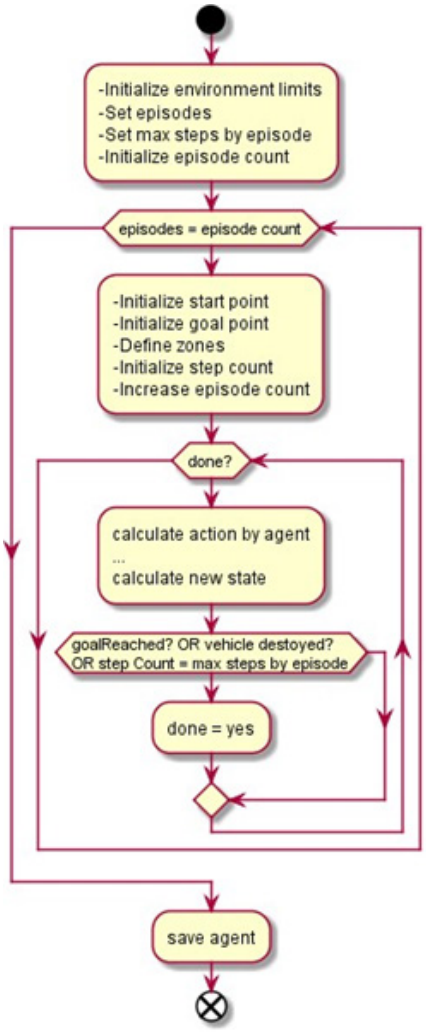


*Figure 2: Algorithm activities*

reach upon a final state condition for each episode, then return to the second step multiple times until to complete all episodes.

Finally, in the training mode saves the agent configuration to be used in other experiments.

### *Neural network*

The neural network topology consists of an input layer to which the environmental observations that are available to the agent are assigned, plus two hidden layers of 256 elements in order to create a system of relative complexity with the ability to learn to multiple environments and to generalize. Finally, the neurons of the output layer indicate which is the action to be performed on the environment.

Using the topology shown in figure 3 six different neural networks were generated by varying the activation functions of the first and second layers as listed in table 1 below.

The Linear function consists of a function that shows the same value of the input as the output, the output range is from minus infinity to plus infinity.

$$f(x) = x \tag{4}$$

The ReLU (Rectified Linear Unit) function consists of a function by parts in which the output is equal to the input when it is positive, otherwise the output is zero, so the output range goes from zero to infinity.

$$f(x) = \begin{cases} 0 \text{ if } x \leq 0 \\ x \text{ if } x > 0 \end{cases} \tag{5}$$

The Sigmoidal function consists of a nonlinear curve in which the input variations slightly affect the output as the output range is from 0 to 1.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{6}$$

The Tanh(hyperbolic tangent) function consists of a nonlinear curve similar to the sigmoidal function however, the output range is between -1 and 1.

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{7}$$

## *RESULTS*

### *General results*

Once the training of the twelve agents during 5000 episodes has been completed, the results are shown in Figure 4, where the rewards obtained throughout the train-

*Table 1: Neural network layers configurations*

| | Activation function of the first layer | Activation function of the second layer |
|---|---|---|
| Linear - Linear | Linear | Linear |
| Linear - Sigmoidal | Linear | Sigmoidal |
| RELU - RELU | RELU | RELU |
| Sigmoidal - Sigmoidal | Sigmoidal | Sigmoidal |
| Sigmoidal - Tanh | Sigmoidal | Tanh |
| Tanh - Tanh | Tanh | Tanh |

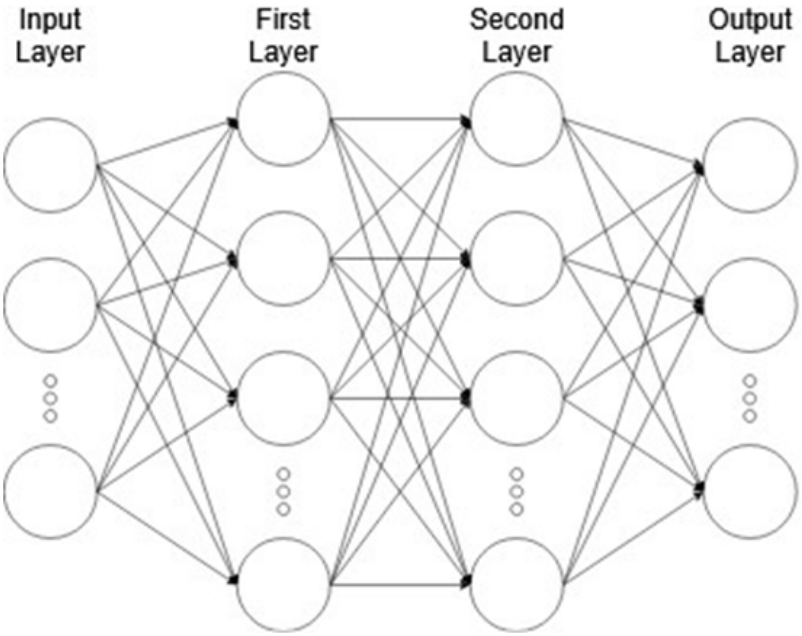


*Figure 3: Neural network diagram*

ing can be observed. In the left column of the figure the results of the DQN agents are observed, in the central column those of the DDQN agents and finally in the right column a comparison between the best DQN agent and the best DDQN. In turn, the first row shows the reward achieved, the second row the length in steps and the last row shows the average time used by each step during the episode.

In the previous figure it is observed that the agents configured with non-linear output functions have better behavior regarding training and computational cost. The first for reaching better results in the number of rewards and reaching a relatively stationary state in less episodes, making at least 3000 of these dispensable; and the second, because these agents use less length in steps even when the times are generally similar. It is also recognized that although the results of the best DQN and DDQN agents converge towards the same window of reward quantity, the one with the highest reward is the DDQN agent with sigmoidal activation functions for the first layer and hyperbolic tangent(tanh) for the second, followed by the DQN agent with the same activation functions, the last one obtaining the results in lower average times per step.

Based on the results, it is determined that the best agent is the one that achieves the highest reward at the end of the 5000 episodes. Therefore, the selected DDQN agent with the highest number of episodes is trained.

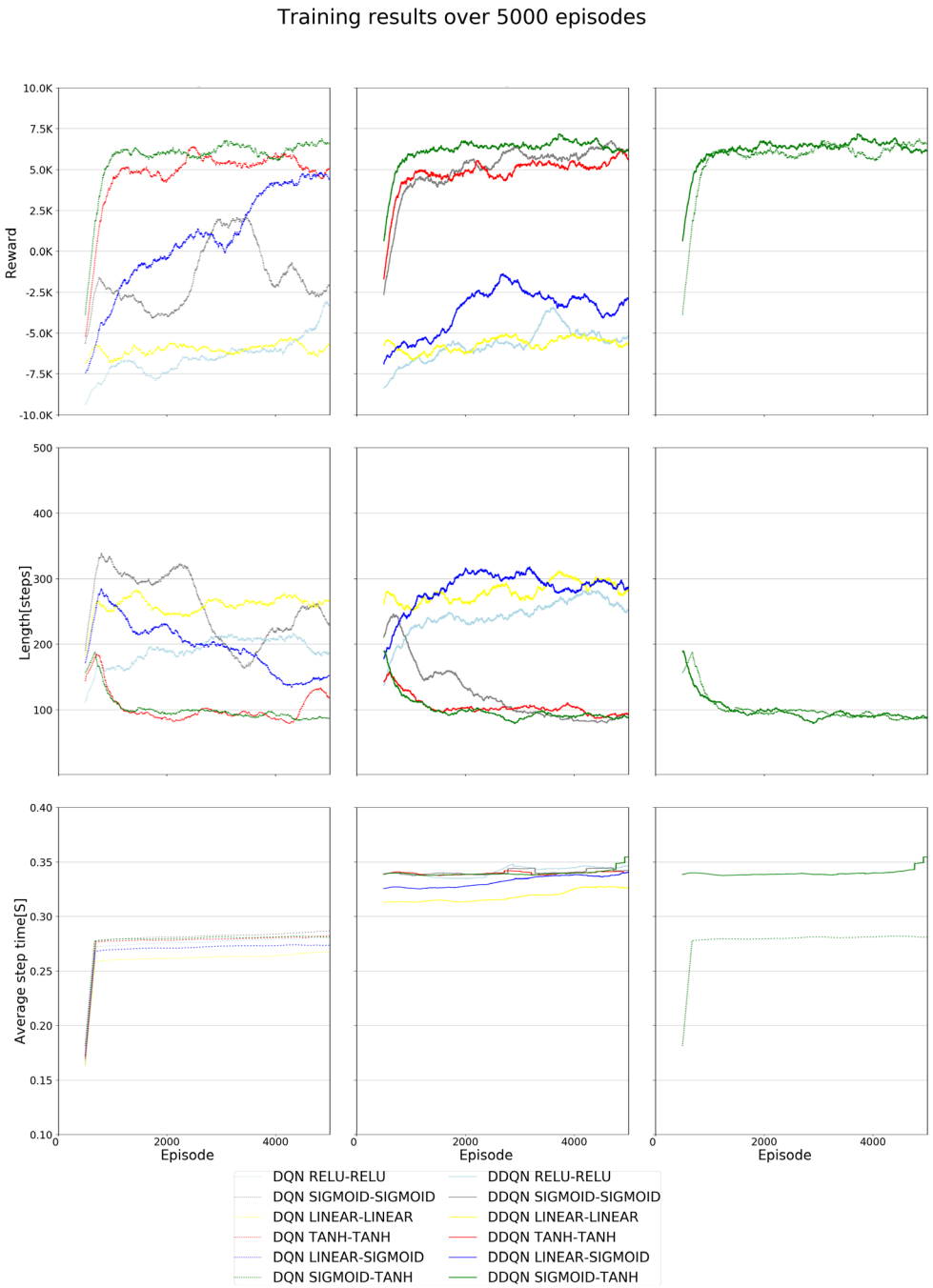


*Figure 4: Comparing results over different configuration of agents*

### *Selected system*

The training of the best agent was performed with 100.000 episodes, a fact represented in Figure 5, which shows the reward and the length in steps for each episode. The grey line for each case shows the evolution made in each one of the episodes, and given its high variation among the episodes it is smoothed made an average every 1000 episodes to obtain the blue line, with which it is appreciated that the greater rate of learning was achieved until the 20000 episode, between 20000 and 40000 the rate of learning decreases and from there on it stays stable with oscillations. Similarly, the length in steps increases rapidly at the beginning until a peak from which it decreases until it remains constant.

In order to validate the selected system, an experiment of 100 episodes is performed using the agent without any learning action. Its results are shown in figure 6, with which it is certified that the path planning system allows to reach the destination in 89% of the occasions, and given the randomness of each one of the episodes, the step length is kept in constant variation between 0 and 150.

In addition, Figure 7 shows two results of the validation process in order to observe the trajectories obtained in the episodes executed after the training. By means of these, it is observed how the object reaches its objective by generating a path that avoids obstacles in the forbidden zone and warning.

### *CONCLUSIONS*

The effectiveness of the mobile robot in reaching its target is increased by the use of an extra neural network to assist the DQN reinforcement learning process. This is because, as exposed [11, 17], Qlearning is an algorithm that requires the computational cost load to improve, and neural networks, as exposed [2, 9] and validated by this study, have the capacity to provide it. It is observed in the validation that the Double DQN method obtains better results than those obtained with DQN, nevertheless, when comparing the time used in training by each step of the episode it is found that the one used by DQN is 15.63% shorter, this as a consequence of which the increase of the amount of parameters of the DDQN method, besides

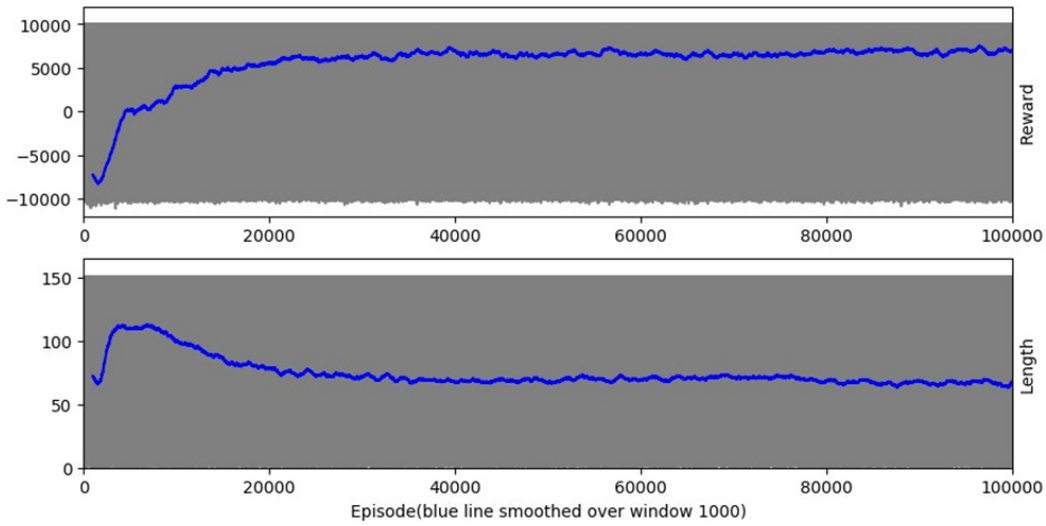


*Figure 5: Results for training episodes*

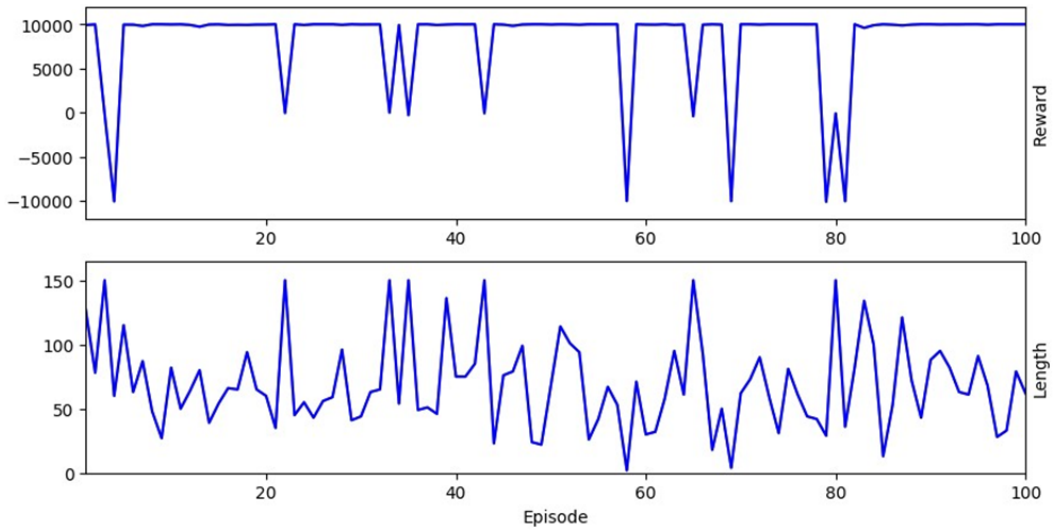


*Figure 6: Results for test episodes*

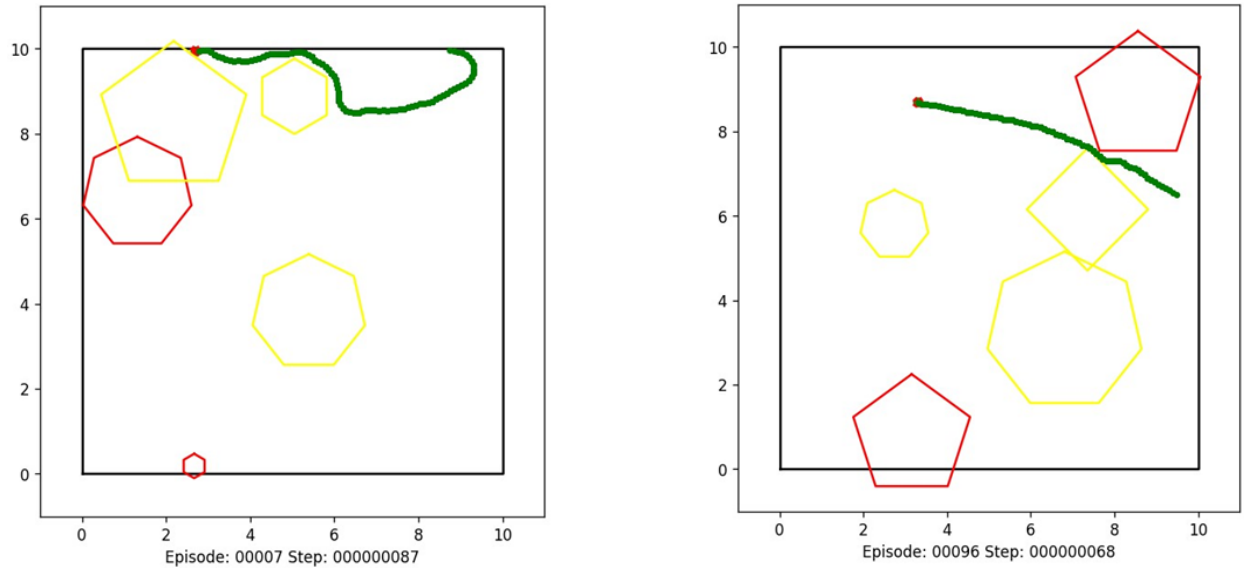


*Figure 7: Results of the path planning for episodes 7 (left) and 96 (right)*

equipping it with more capacity, evidently also increases the complexity of the system.

With respect to the results obtained from the different path planning systems according to the activation functions of the neuronal networks, it is concluded that the non-linear functions present a better performance in achieving the objective compared to the linear functions, this being based on the plasticity that the systems based on computational intelligence possess, as can be observed in [2, 3, 4], and that it is supported with the use of output responses by means of non-linear behavioral functions.

The result is generally acceptable since the target point was reached in 89% of the episodes carried out in the validation experiment; however, it is necessary to establish a balance between computational cost and system efficiency, which will depend on the complexity of the environment and the neural network topology, leaving as future work the study of the use of neural networks with a greater number of layers and/or fewer neurons to minimize the dimensionality of the system while taking advantage of the computational intelligence resource.

## ACKNOWLEDGMENT

The current work was developed with the support of CECAD (High-performance computing center of the Universidad Distrital Francisco José de Caldas) who provided the technological platform required to perform the training and test of the neural networks. The name of the project is "Plataforma estratosférica de vuelo autónomo Sabio Caldas". The code of the project is 3307348415.

## REFERENCES

1. Praveen Kalla et al. “Coordinate Reference Frame Technique for Robotic Planar Path Planning”. In: *Materials Today: Proceedings* 5.9, Part 3 (2018). Materials Processing and characterization,16th – 18th March 2018, pp. 19073–19079. issn: 2214-7853. doi: 10.1016/j.matpr.2018.06. 260. url: http://www.sciencedirect.com/science/article/pii/S221478531831383X.
2. Chaymaa Lamini, Said Benhlima, and Ali Elbekri. “Genetic Algorithm Based Approach for Autonomous Mobile Robot Path Planning”. In: *Procedia Computer Science* 127 (2018). PRO-CEEDINGS OF THE FIRST INTERNATIONAL CONFERENCE ON INTELLIGENT COMPUTING IN DATA SCIENCES, ICDS2017, pp. 180–189. issn: 1877-0509. doi:10.1016/j.procs.2018.01.113. url: http://www.sciencedirect.com/science/article/pii/S187705091830125X.
3. Thi Thoa Mac et al. “Heuristic approaches in robot path planning: A survey”. In: *Robotics and Autonomous Systems* 86 (2016), pp. 13–28. issn: 0921-8890. doi: 10.1016/j.robot.2016.08. 001. url: http://www.sciencedirect.com/science/article/pii/S0921889015300671.
4. Azzeddine Bakdi et al. “Optimal path planning and execution for mobile robots using genetic algorithm and adaptive fuzzy-logic control”. In: *Robotics and Autonomous Systems* 89 (2017), pp. 95–109. issn: 0921-8890. doi: 10.1016/j.robot.2016.12.008. url: http://www.sciencedirect.com/science/article/pii/S0921889016302512.

5. Farhad Bayat, Sepideh Najafinia, and Morteza Aliyari. “Mobile robots path planning: Electrostatic potential field approach”. In: *Expert Systems with Applications* 100 (2018), pp. 68–78. issn: 0957-4174. doi: 10.1016/j.eswa.2018.01.050. url: http://www.sciencedirect.com/science/article/pii/S0957417418300630.
6. Julien Moreau et al. “Reactive path planning in intersection for autonomous vehicle”. In: *IFAC-PapersOnLine* 52.5 (2019). 9th IFAC Symposium on Advances in Automotive Control AAC 2019, pp. 109–114. issn: 2405-8963. doi: 10.1016/j.ifacol.2019.09.018. url: http: //www.sciencedirect.com/science/article/pii/S2405896319306378.
7. Yi-Wen Chen and Wei-Yu Chiu. “Optimal Robot Path Planning System by Using a Neural Network-Based Approach”. In: *Proceedings of 2015 International Automatic Control Conference (CACS)* (2015), pp. 85–90.
8. Valeri Kroumov and Jianli Yu. “Neural Networks Based Path Planning and Navigation of Mobile Robots”. In: *Recent Advances in Mobile Robotics* (2011), pp. 173–190.
9. Keyu Wu et al. “TDPP-Net: Achieving three-dimensional path planning via a deep neural network architecture”. In: *Neurocomputing* 357 (2019), pp. 151–162. issn: 0925-2312. doi: 10.1016/j.neucom.2019.05.001. url: http://www.sciencedirect.com/science/article/ pii/S092523121930606X.
10. Md. Arafat Hossain and Israt Ferdous. “Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique”. In: *Robotics and Autonomous Systems* 64 (2015), pp. 137–141. issn: 0921-8890. doi: 10.1016/ j.robot.2014.07.002. url: http://www.sciencedirect.com/science/article/pii/S0921889014001274.
11. Ee Soong Low, Pauline Ong, and Kah Chun Cheah. “Solving the optimal path planning of a mobile robot using improved Q-learning”. In: *Robotics and Autonomous Systems* 115 (2019), pp. 143–161. issn: 0921-8890. doi: 10.1016/j.robot.2019.02.013. url: http://www.sciencedirect.com/science/article/pii/S0921889018308285.
12. Adam C Parry and Raul Ordonez. “Intelligent Path Planning with Evolutionary Computation”. In: *AIAA Guidance, Navigation, and Control Conference* August (2010).
13. Alessandro Gasparetto et al. “Path Planning and Trajectory Planning Algorithms: a General Overview”. In: *Motion and Operation Planning of Robotic Systems*. September 2017. 2015. Chap. 1, pp. 3–27. isbn: 9783319147055.
14. Yandun Aracely and Sotomayor. Nelson. “Planeacion y seguimiento de trayectorias para un robot movil”. In: (2012).
15. Chen Chen et al. “A knowledge-free path planning approach for smart ships based on reinforcement learning”. In: *Ocean Engineering* 189 (2019), p. 106299. issn: 0029-8018. doi: 10.1016/j.oceaneng.2019.106299. url: http://www.sciencedirect.com/science/article/pii/S0029801819304706.
16. Richard S. Sutton and Andrew G Barto. R*einforcement Learning: An Introduction*. 2nd ed. 2018, p. 552. isbn: 978-0262039246.
17. Ding Ding et al. “Q-learning based dynamic task scheduling for energy-efficient cloud computing”. In: *Future Generation Computer Systems* 108 (2020), pp. 361–371. issn: 0167-739X. doi: 10.1016/j.future.2020.02.018. url: http://www.sciencedirect.com/science/article/ pii/S0167739X19313858.
18. Gyeeun Jeong and Ha Young Kim. “Improving financial trading decisions using deep Q-learning: Predicting the number of shares, action strategies, and transfer learning”. In: *Expert Systems with Applications* 117 (2019), pp. 125–138. issn: 0957-4174. doi: 10.1016/j.eswa.2018.09.036. url: http://www.sciencedirect.com/science/article/pii/S0957417418306134.
19. Martin Riedmiller. “Neural fitted Q iteration - First experiences with a data efficient neural Reinforcement Learning method”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 3720 LNAI (2005), pp. 317–328. issn: 03029743.
20. Sebastian Thrun and Anton Schwartz. “Issues in Using Function Approximation for Reinforcement Learning”. In: *Proceedings of the 4th Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum* (1993), pp. 1–9.
21. Hado Van Hasselt. “Double Q-learning”. In: *Advances in Neural Information Processing Systems*. January 2010. 2010.
22. Hado Van Hasselt. *Insights in Reinforcement Learning: Formal Analysis and Empirical Evaluation of Temporal-Difference Learning Algorithms*. 2011, pp. 1–282. isbn: 9789039354964.


*Paper submitted: 03.05.2020.*
*Paper accepted: 22.09.2020.*