A Trajectory Simulation Approach for Autonomous Vehicles Path Planning using Deep Reinforcement Learning

Jean Phelipe de Oliveira Lima
Superior School of Technology, Amazonas State University,
Manaus-AM, Brazil.
ORCID: https://orcid.org/0000-0002-5861-9928
Email: jpdol.eng16@uea.edu.br

Raimundo Correa de Oliveira
Superior School of Technology, Amazonas State University,
Manaus-AM, Brazil.
ORCID: https://orcid.org/0000-0002-5428-8762
Email: rcoliveira@uea.edu.br

Cleinaldo de Almeida Costa
Superior School of Health, Amazonas State University,
Manaus-AM, Brazil.
ORCID: https://orcid.org/0000-0001-8400-5543
Email: cleinaldocosta@uea.edu.br

Abstract

Autonomous vehicle path planning aims to allow safe and rapid movement in an environment without human interference. Recently, Reinforcement Learning methods have been used to solve this problem and have achieved satisfactory results. This work presents the use of Deep Reinforcement Learning for the task of path planning for autonomous vehicles through trajectory simulation, to define routes that offer greater safety (without collisions) and less distance for the displacement between two points. A method for creating simulation environments was developed to analyze the performance of the proposed models in different difficult degrees of circumstances. The decision-making strategy implemented was based on the use of Artificial Neural Networks of the Multilayer Perceptron type with parameters and hyperparameters determined from a grid search. The models were evaluated for their reward charts resulting from their learning process. Such evaluation occurred in two phases: isolated evaluation, in which the models were inserted into the environment without prior knowledge; and incremental evaluation, in which models were inserted in unknown environments with previous intelligence accumulated in other conditions. The results obtained are competitive with state-of-the-art works and highlight the adaptive characteristic of the models presented, which, when inserted with prior knowledge in environments, can reduce the convergence time by up to 89.47% when compared to related works.

Keywords: Autonomous Vehicles; Deep Reinforcement Learning; Path Planning; Trajectory Simulation.

1. Introduction

The evolution of mobile robotics ensures the creation of technology for automating everyday tasks [1], from robots for domestic use, as presented in [2], often based on reactive control [3], to autonomous land vehicles [4], air vehicles [5] or water vehicles [6] and self-guided vehicles in Industry 4.0 [7], based on advanced control and automation techniques or artificial intelligence. The fundamental task performed by an autonomous vehicle, regardless of the environment in which it operates, is the ability to move around [8], whether in a controlled or an uncontrolled environment. An autonomous vehicle must be able to receive data from the environment, process it for decision-making, and take action through actuator devices.

According to [9], the main challenges offered regarding the development of autonomous vehicles are: Path Planning and Collision Avoidance. Path Planning is the system's ability to decide a good route to be taken by the vehicle [4]. The quality of the chosen path can be assessed through metrics such as: travel time; path distance; and the number of collisions [10]. Collision Avoidance [11] refers to the necessity for the determined route to be free of collisions with possible obstacles. Furthermore, being essentially important in this context, as it represents the safety of a path, Collision Avoidance is directly related to Path Planning, so there is a demand for a system able to find a satisfying route in terms of safety and distance [12].

Figure 1 shows a high-level architecture for an autonomous vehicle, in which its main modules are observed. Sensing & Mapping and Acting are interface modules with the environment. Sensing & Mapping is the autonomous vehicle module that receives inputs from the environment, usually through sensor networks [13], aiming to process them to be understood, by the vehicle, of the environment in which it operates, in other words, to perform mapping and auto localization concerning the environment. For this processing, the Simultaneous Localization and Mapping (SLAM) technique [14] is shown to be robust in several situations using different forms of interaction with the environment. In [6], for example, SLAM is applied to an autonomous underwater vehicle that senses the environment from sonar signals, whilst in [15] SLAM is implemented using signals and video camera images.

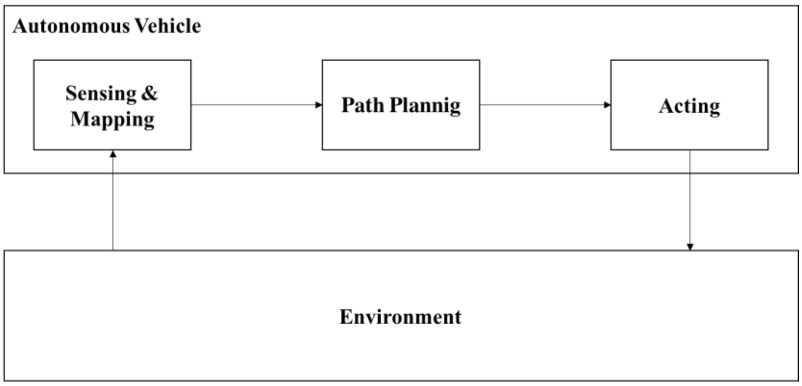

Figure 1: A High-level architecture of an Autonomous Vehicle System.

Nevertheless, the Acting module is responsible for applying to the environment the actions decided by the autonomous vehicle, such as traction and rotation of the engine and emergency stop, for example, through peripherals called Actuators. In this segment, many works present applications of techniques based on the Theory of Classical and Modern Control [16], as in [17], which presents a system of variation of buoyancy based on the Proportional Integral Derivative controller (PID) applied to robotics autonomous navy, and in [18], which presents a PID controller for electric vehicle traction motors. In various recent works, these controllers are equipped with artificial intelligence, as in [19], [20] and [5], in which the authors use, respectively, artificial neural networks with online training, genetic algorithms and Fuzzy inference to optimize the parameters of PID controllers, which add an adaptive feature to the controller, in which the parameters are adjusted in real-time, thus increasing the robustness of the control system for possible disturbances in the input variables.

The Path Planning module may be compared to an autonomous vehicle data processing center. It is the module responsible for planning and (re) defining, in real-time, a route to be determined by the vehicle, from the mapping data, and transmitting this information to the Acting module. For this activity, artificial intelligence techniques, especially reinforcement learning, have been applied, as is the case in [21], in which models based on two Deep Reinforcement Learning techniques are presented called Deep Deterministic Policy Gradient (DDPG) and Multiple Experience Pools DDPG (MEP-DDPG) for planning autonomous movement of aerial unmanned vehicles (AUVs). Likewise [22] presents the application of DDPG for planning land vehicle routes.

Therefore, a Path Planning method for autonomous vehicles based on Trajectory Simulation using Deep Reinforcement Learning is proposed. Given a mapped environment, Artificial Neural Networks will be used to define the quality of the actions to be taken by an autonomous vehicle. This decision-making process will be based on rewards obtained from simulations of trajectories carried out in the environment, it means that the model becomes capable of evaluating actions based on experience acquired by simulations in the environment. As the intelligent model and the implementation pipeline of the artificial intelligence system is based on reinforcement learning, the idea is that simulations are accomplished until the model finds the route that offers the best rewards, which represent, for the model, safety and cost conditions until reaching the predetermined objective, thus performing Collision Avoidance and Path Planning, respectively.

Section 1 presented the Introduction to the theme, the concepts and the problem addressed by this work, as well as a contextualization concerning the related works observed in the literature. Section 2 presents the materials and methods employed for the development, in which it describes, minutely, the machine learning approach used as well as the components created to conduct the experiments and presents the methods of choosing the architecture of the smart models tested and evaluation of the results obtained. Section 3 presents and analyzes the results obtained regarding the proposed method and compares the performance of the system to related research. Lastly, in Section 4, final considerations and perspectives for future work are presented.

2. Materials and Methods

This section describes the materials and methods used for the development and analysis of the experiments conducted. The subsections contain: (2.1) Description of Machine Learning Approach, which indicates an overview of how machine learning was applied in this work; (2.2) Reinforcement Learning Components, which presents in details all the components that integrate the reinforcement learning system; (2.3) Neural Network Architecture, which exposes the methodology of defining the neural network architecture used; and, eventually, (2.3) Model Evaluation, which informs how the performance evaluation of the models proceeds for the task in question.

2.1 Description of Machine Learning Approach

This work presents Reinforcement Learning (RL) [23] with the task of simulating trajectory for autonomous vehicles, with the purpose of planning routes in previously mapped environments. The experiments were conducted in a computer-simulated environment. Figure 2 presents the architecture of a RL system where the system components interaction can be conceived.

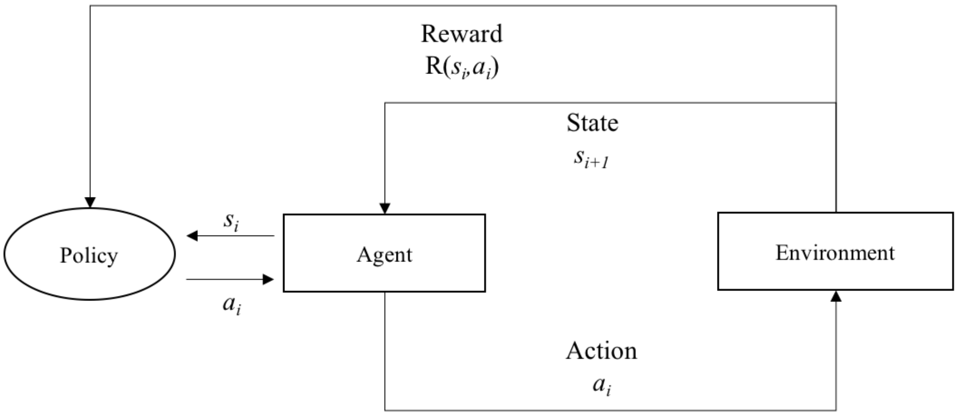

Figure 2: Reinforcement Learning System architecture.

The interaction dynamics of the components of a RL system, as illustrated by Figure 2, consists of: an Agent (actor of the system) that from a state s_i (situation proposed by the environment), chooses an Action a_i (an activity that the Agent can perform to interact with the Environment) based on a Policy (decision-making strategy) and, according to this interaction with the environment, the Agent assumes a new State s_{i+1} and the policy is updated by a Reward $R(s_i, a_i)$ received by the action taken from the previous state, which can be positive or negative (also called Penalty). Therefore, an epoch of reinforcement learning ends. Thereafter the new state s_{i+1} becomes the current state s_i and the process is repeated until learning is reached.

In the context of this work, the Agent is represented by an autonomous vehicle, which has a constant velocity and its possible actions are associated with the direction of the route. The simulation environment created computationally presents arbitrary obstacles and a determined point of origin and destination. Whether a contact of the agent with the environment obstacles occurs, it signifies the agent achieves an obstacle state, thereby receiving a negative reward, as well as when happens excessive steps before the destination. Hence, the agent's purpose is to establish a trajectory between the points of origin and destination with the greatest possible reward, promptly and without collisions. The decision policy is based on the Deep Q-Learning approach [24], which considers the classic Q-Learning algorithm [25] in which, from the Q function:

$$Q(s,a) = R(s,a) \ + \ \gamma \cdot (\delta(a,s,s') \cdot max_{a'}(Q(s',a'))) \qquad (1)$$

originated from the general equation from Bellman [26], a table of qualities is created that relates the agent's actions to the states to be achieved through the defined actions by a Markov Decision Process [27]. Concerning the function, it assumes: s the current state; a the executed action to reach the state s; R the rewards function; γ discount factor; and δ the state transition function (which defines the probability of reaching the state s', once it was in s and a was executed, having $P(s'|(s,a))$. In other words, the Q equation may represent, mathematically, Figure 2. Deep Q-Learning, otherwise, approximates the quality values for each action from Artificial Neural Networks [28] (ANN), thus it creates an intelligent model capable of deciding the actions to be taken by the agent in the environment.

2.2 Reinforcement Learning Components

The components of the Reinforcement Learning System presented in this work will be fully detailed in the following topics, divided respectively into: Environment and States; Agents and Actions; Rewards; and Policy and Exploitation / Exploration.

2.2.1 Environment and States

The work was performed in a graphical environment developed in Python version 3.7.4 through the Kivy framework, in its version 1.11.1, which uses OpenGL version 4.5 resources as a back-end. An environment may be abstracted as a 600x800 dimension matrix, so 600 rows and 800 columns. Every position in this matrix represents a state, which may be achieved by the agent.

A state may contain an obstacle or not. Considering the idea is to simulate a real environment, which does not have a specific pattern of obstacle disposition, it is settled to randomly generate obstacles following some rules of degrees of difficulty, as it appears in [21]. Therefore, the RL model will be prepared and assessed in environments with degrees of difficulty: Easy; Medium; Difficult. These difficulties were established in relation to the number of obstacles and their respective sizes, whilst their position was established utterly casually. Toward this purpose, the lower and upper limits of quantity and size of obstacles were empirically set, which assisted as references for random generation. Table 1 presents these intervals.

Table 1: Intervals of values of quantity and size of obstacles for each degree of difficulty.

Difficulty Level	Number of Obstacles		Size of Obstacles	
	Lower Limit	Upper Limit	Lower Limit	Upper Limit
Easy	10	20	10	60
Medium	20	30	20	80
Hard	30	40	30	100

Thus, from these intervals, the generation of obstacles is arranged:

1. The difficulty level of the environment is defined;
2. A random value $n \in [n_i, n_s]$ is generated, with n_i, n_s, respectively, the lower and upper limit of the number of obstacles according to the degree of difficulty. This process will determine the number of obstacles in the environment;
3. Repeat steps 4 and 5 n times;
4. A random value $s \in [s_i, s_s]$ is generated, being s_i, s_s, respectively, lower and upper limit of the number of obstacles according to the degree of difficulty. This process determines one of the n obstacles that will be created;
5. An obstacle is created from a point of origin $p_o = (x_o, y_o)$ to an end point $p_f = (x_f, y_f)$, in that x_o, y_o and x_f, y_f are randomly defined, with $x_o, x_f \in [0{,}800]$, $y_o, y_f \in [0{,}600]$, and $(x_f - x_o)^2 + (y_f - y_o)^2 = s^2$, it means, p_f distances s from p_o.

Also, the regions of origin and destination of the agent were defined, being located in the upper and lower central part of the environment. As the proposed task is to obtain a trajectory, and not to find a final state, as is the case with most RL systems, it was decided to alternate the regions of origin and destination, in other words, when the agent assumes a state in the region of destination, this becomes the region of origin and the region of origin becomes the new region of destination, so the agent travels several routes between the two regions until it discovers the one that offers the greatest possible reward.

2.2.2 Agents and Actions

The agent simulates a vehicle that can move around in the environment. It has constant velocity and degrees of freedom regarding possible directions, choosing to go forward or perform a 45^o inclination both to the right and to the left. These operations define the set of actions that the Agent may take, denoted by $A = \{0^o, 45^o, -45^o\}$.

Figure 3 exemplifies some states that the agent may be in, the possible actions to take and the resulting states from each action. The colored circles simulate sensors that identify possible states to reach for each action.

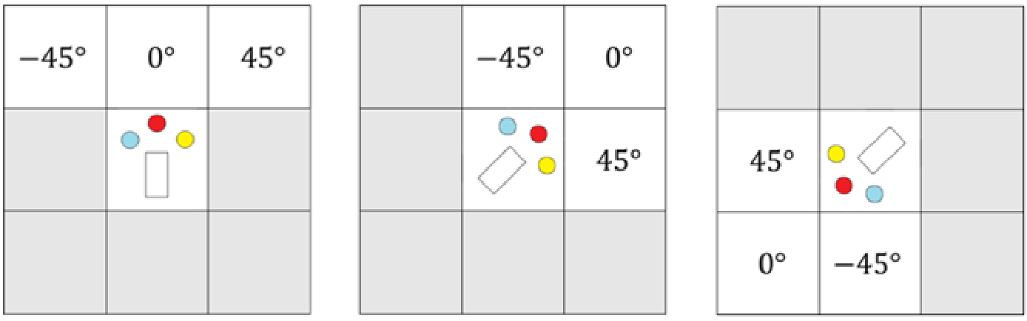

Figure 3: Examples of possible actions to take by the agent and the respective resulting states for each action.

2.2.3 Rewards

Possible cases have been raised based on the actions occasionally taken by the agent. One case represents the situation of the agent regarding its aim of finding a rapid trajectory and without collisions between the defined points of origin and destination. According to the proposed task, the importance of each case is rated as: Strong Positive, when the case has strong representativeness in favor of the wanted purpose; Weak Positive, when the case has weak representativeness in favor of the wanted purpose; Strong Negative, when the case has strong representativeness contrary to the purpose; and Weak Negative, when the case has weak representativeness contrary to the purpose.

The reward values were defined in the range of $[-1,+1]$ from tests based on empirical results presented in [29]. For cases with a Strong Positive rating, a reward of $+1$, was assigned, for cases with a Weak Positive rating, a reward of $+0.2$, for cases with a Strong Negative rating, a reward of -1, and for cases with a Weak Negative rating, a reward of -0.2. Table 2 presents the raised cases and their respective ratings and rewards.

Table 2: Table of rewards based on the cases in which the agent may be in the environment.

Case	Rating	Reward
Collision with Obstacles	Strong Negative	-1
Collision with limits of the environments	Strong Negative	-1
Distance to destination has decreased since the last iteration	Weak Positive	+0.2
Distance to destination has not decreased since the last iteration	Weak Negative	-0.2

It is observed from 2, the “Strong Positive” rate was not attributed to any case, because as the assignment is to find a trajectory between the regions of origin and destination, to offer the agent a very high reward for a proximity to the destination can create a tendency in the model, in addition to a misleading analysis in the rewards graph, which would present very high values even before a supposed ideal path was found. Another classic option would be to repay the agent with high rewards when the destination region is reached, which is appropriate when the task of RL is to discover a state, which is not the case, therefore, doing so, an error in the analysis of the rewards graph would also occur: average high rewards even if Weak Negative cases happened, so the model would have its learning influenced by a high reward having neglected important small penalties.

Hence, it is reasonable to affirm the task is dictated by negative rewards and the agent holds the main purpose of not to suffer penalties, which is suitable for the model, since the importance of not having collisions on the route is greater than the one to find the shortest path. In this case, the analysis of the rewards graph will happen explicitly: high negative values will appear when there are collisions, otherwise, the values should fluctuate around zero, tending to stabilize as the proximity to the ideal trajectory happens, that is, the shortest path without collisions.

2.2.4 Policy and Exploitation / Exploration

The decision-making policy used is based on artificial neural networks of the Multilayer Perceptron (MLP) type [30] with online training [31], that is, the weights are continuously adjusted as experience is being acquired by the agent. This experience is a result of the input interaction, processing and neural network output. At the end of each interaction, also called a step, the reward value is presented to the network that performs weight adjustment in order to maximize the reward values, reinforcing the weights of the neural connections that have the greatest influence on an output that led to a positive reward and penalizing the connections with the greatest contrary influence, thus realizing an approximation to the Q function, presented in equation (1). Figure 4 illustrates this learning process of the neural network from the reinforcement resulting from the actions taken.

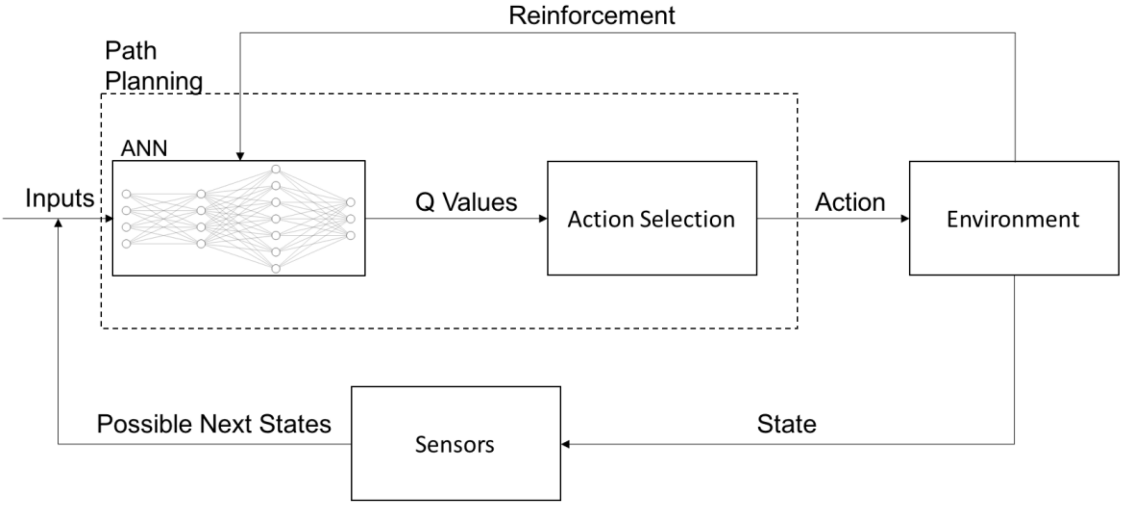

Figure 4: Process of online training of the neural network from the reinforcement resulting from the actions taken.

ANN entries represent the situation the agent is in at a certain time. Thus, the network receives data from the sensors, which correspond to the possible states to reach through an action, and receives the rotation of the agent concerning the environment, which can be seen in Figure 3, in which the agent appears with an inclination $0^o, 45^o$ and 135^o, respectively from left to right. The network processes this data in the hidden layers, and returns, in the output layer, the approximation to the Q value for each action, it means, for each neuron in the output layer, and the neuron with the highest Q value corresponds to the suggested action for the exact instant.

An important issue in Reinforcement Learning is the Exploitation / Exploration dilemma [32]. Exploitation is when decision-making is fulfilled from experience, that is, from the learning the model has had until then, while Exploration consists of decision-making, usually random, aiming to increase the model's experience. The use of neural networks, which weights are started randomly, means that, until a certain moment, decisions are made with the intention of exploration, since the network does not have, or has little, prior knowledge regarding the environment. When experiences are created by the model, there is a risk that decision-making will become tendentious, which means the model always chooses to follow paths already taken, even if these are not the shortest ones. Given the dilemma, it was decided to manage the dynamics of Exploitation / Exploration randomly with probabilities of occurrence of 0.9 and 0.1 respectively, which is illustrated by Figure 5 through a Probabilistic Finite Automaton (PFA) [33].

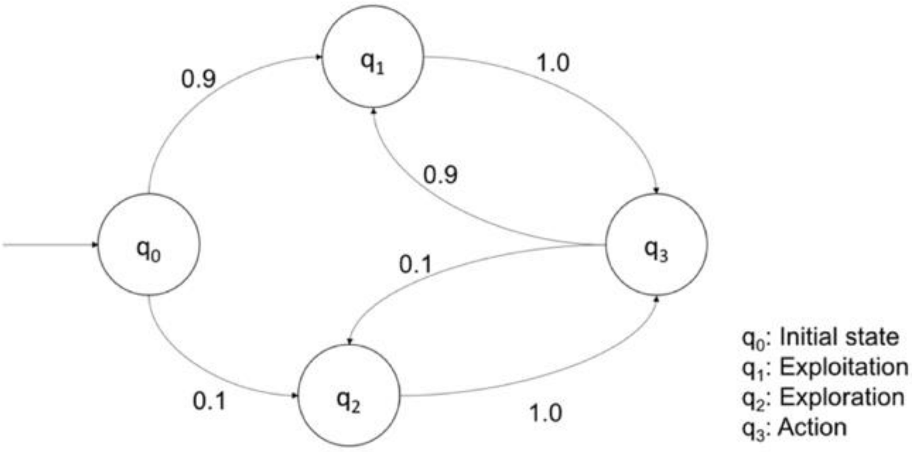

Figure 5: Probabilistic Finite Automaton of Exploitation / Exploration.

Figure 5 indicates that, from the initial state q_0, the next state reached is the decision-making Exploitation or Exploration, represented respectively by q_1 and q_2, with the Exploitation state being reached in 90% of the cases, in which the neural network decides which action to take, and the Exploration state in 10% of the cases, in which the action is decided randomly. After the decision-making stage is completed, that is, after states q_1 or q_2, the state reached is q_3, that corresponds to the state in which the action is performed. Once the determined action is performed, one of the decision-making states q_1 or q_2 is reached again respecting the same probabilities as before. This cycle defines the policy used for RL.

2.3 Neural Network Architecture

The architecture of the multi-layer Perceptron network applied for the agent's decision-making process was established through a grid search, as described below, to optimize the search for its best parameters and hyperparameters. The grid search consists of training all possible combinations of parameters and hyperparameters previously established [34]. For this, an environment of each degree of difficulty was generated and each network generated was used, without prior learning, in the agent's policy that was inserted in the environments. The final metric used to describe the best network architecture was the average, for the performance of the models in the 3 environments, from the number of steps to the convergence of the rewards graph.

The pre-defined parameters and hyperparameters for the grid search were: number of hidden layers, number of neurons per hidden layer, batch size, learning rate, activation function and optimizer. The number of hidden layers was fixed at 2, since, with two hidden layers, a neural network can implement any function [30]. The number of hidden neurons was obtained by the Geometric Pyramid Rule (GPR) [35], given by:

$$N_h = \lceil \alpha \cdot \sqrt{N_i \cdot N_o} \rceil \tag{2}$$

where N_h is the total number of hidden neurons, N_i the number of neurons in the input layer, which corresponds to the number of predictive attributes, N_o the number of neurons in the output layer, which corresponds to the number of classes, and α a constant arbitrary. All combinations of two layers were admitted, formed by the total of hidden neurons obtained by GPR when $\alpha = 1;\ 1.5;$ and 3. The batch size was $16, 24$ and 32. The determined learning rates were 0.0001 and 0.0005. While the tested optimizers [37] were the Adam algorithm; Stochastic Gradient Descent (SGD); and Limited-Memory Broyden–Fletcher–Goldfarb–Shanno (LBFGS) [36]. Ultimately, the activation functions were ReLU:

$$ReLU(x) = max(0, x) \tag{3}$$

Sigmoidal:

$$Sig(x) = \frac{1}{1+e^{-x}} \tag{4}$$

Hyperbolic Tangent:

$$tanh(x) = \frac{senh(x)}{cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{5}$$

and Identity:

$$Ident(x) = x \tag{6}$$

2.4 Model Evaluation

The evaluation metric used to assess the models is the Rewards Graph [38], which consists of a rewards curve according to the steps, where it is possible to observe, mainly, the learning evolution in each environment and the number of steps until convergence, that indicates the number of interactions required till the model finds a satisfactorily short and collision-free path. The evaluation of the models will be performed in two phases: Isolated Evaluation and Incremental Evaluation, which will be detailed in the following topics.

2.4.1 Isolated Evaluation

The isolated assessment consists of analyzing the performance of the models in each environment without any prior knowledge. Thus, the testing routine will be displayed as follows: for each degree of difficulty, 100 environments will be generated, according to the proposed methodology, and in each environment, an agent without prior knowledge will be included. Hence, the final performance metric will be the average reward curves of the 100 models referring to the environments of each degree of difficulty.

2.4.2 Incremental Evaluation

The incremental assessment consists of analyzing the performance of the models in environments from the lowest to the highest level with an accumulation of knowledge between the environments. Therefore, the testing routine will be performed as follows: for each degree of difficulty 100 environments will be generated, according to the proposed methodology, then an agent will be inserted in an environment of easy degree, after the convergence of the curve, the same agent will be inserted in an environment of medium degree until convergence, and, finally, inserted in a difficult-grade level.

The final performance metric will be the average curve of the agent's performance in medium environments, holding previous knowledge of an easy environment, and the agent's performance in difficult environments, having preceding data of an average environment. Through these reward curves generated by a model with acquired information, it will be feasible to analyze, in comparison with the results of the isolated assessment, the adaptive nature of the models, that is, when a model with experience is inserted in a new and, by the time, unknown environment.

3. Results and Discussions

This section describes the results achieved from the proposed methodology for simulating the trajectory of autonomous vehicles through deep reinforcement learning. The subsections include the following topics: (3.1) Environments Generation, which presents the results from the proposed method for the creation of RL environments; (3.2) Neural Network Architecture, which presents the best network architecture found from the grid search; (3.3) Isolated Evaluation; and (3.4) Incremental Evaluation, which present the metrics obtained for the respective forms of evaluation of intelligent models.

3.1 Environments Generation

The environments were developed respecting the proposed methodology and divided between the degrees of difficulty: Easy; Medium and Difficult. Figure 6 displays an example of each degree of difficulty from the environments. It is observed that the parameters of quantity and size of obstacles defined for the creation of the environments were properly adjusted, making it clear the levels of difficulty that each environment must intend to the agent. The difficulty increases from left to right, the environment on the left being easy, the middle environment, medium and the right environment, difficult. At the upper and lower edges in the center of each of the environments, the regions of destination and origin of the agent are marked in red, which delimit the path that the agent will take in search of the trajectory that offers it better rewards.

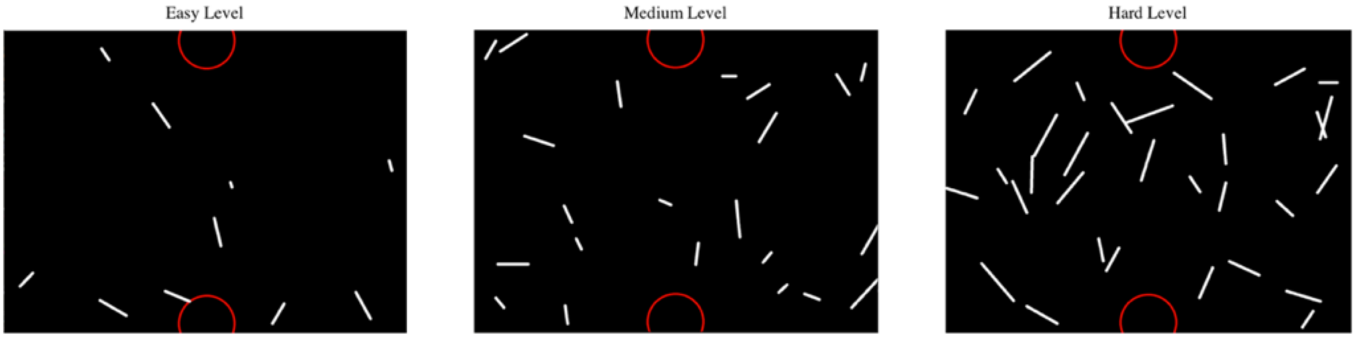

Figure 6: Examples of environments created to perform the proposed RL task.

3.2 Neural Network Architecture

According to the proposed methodology, the neural network adopted for the proposed task was based on the result of a grid search. The network has 4 neurons in the input layer, referring to data from 3 sensors and rotation of the agent concerning the environment, and 3 neurons in the output layer, referring to possible actions to be taken. Amidst these data and with values of $\alpha = 1;\ 1.5$ and 3, from GPR, equation (2), the amounts of hidden neurons $N_h = 2;\ 6$ and 11 were obtained for each value of α. With these values, 16 combinations of neurons are possible, sorted into 2 hidden layers. Table 3 presents the defined parameters and hyperparameters for the grid search.

Table 3: Parameters and Hyperparameters defined for the grid search.

Parameter/ Hyperparameter	Amount of Values	Values
Number of Hidden Layers	1	2
Neurons of Hidden Layers	16	Defined by equation (2)
Batch Size	3	16, 24, 32
Learning Rate	2	0.0001, 0.0005
Optimizer	3	Adam, SGD, LBFGS
Activation Functions	4	ReLU, Sigmoidal, Hyperbolic Tangent, Identity

Therefore, the product of the quantities of values results in the number of combinations between parameters and hyperparameters of neural networks. As a result, 1,152 tests with neural networks were performed. It is noteworthy that in this phase of the experiments, the selection of the neural network was made through the performance of the models in 3 environments, one of each degree of difficulty. These procedures were performed trying to find the combination with the lowest average number of steps until the convergence of the reward charts for the environments where the agents were inserted. Table 4 shows five networks with the best grid search results. Finally, the network ranked first in the grid search was defined as the standard network used in the agent's policy.

Table 4: Results obtained from the grid search.

Classification	Hidden Layers	Activation Function	Optimizer	Learning Rate	Batch Size	Mean of Steps
1	**(4,7)**	**ReLU**	**Adam**	**0.0005**	**16**	**1,998.67**
2	(2,9)	Identity	Adam	0.0005	16	2,005.33
3	(3,3)	Identity	SGD	0.0001	24	2,042.00
4	(4,7)	ReLU	Adam	0.0001	16	2,101.67
5	(2,4)	Sigmoidal	Adam	0.0005	24	2,133.33

3.3 Isolated Evaluation

In the isolated assessment, for each environment, an agent without prior learning was inserted and their rewards graph was analyzed. Altogether, the environments were tested, being divided equally between the three levels of difficulty. Figure 7 shows the performance of the intelligent model proposed for the respective degrees of difficulty according to the average reward curve achieved throughout the conducted experiments.

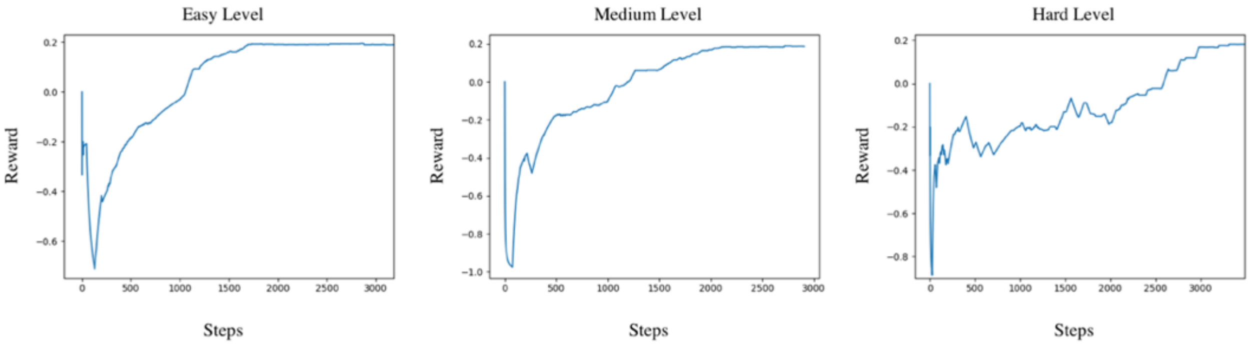

Figure 7: Graphs of average rewards obtained by the models in environments with three degrees of difficulty.

Figure 7 shows a standard behavior in the curves making possible to observe: from the initial moment, the agent has a random behavior, which causes collisions with obstacles, so there are, initially, the presence of some negative peaks. With the increase in steps, the experience is acquired by the agent and the curves tend to converge to a constant reward of 0.2, as expected. This convergent behavior implies that the model has learned from the environment and is capable of drawing a route without penalties, that is, a short path and without collisions.

Furthermore, from Figure 7, it is observed that, in fact, the difficulties of the environment directly affect the number of required interactions until convergence. It is noticed that the graph referring to rewards in environments of easy degree showed convergence around the step 1500, that is, fewer interactions than the graphs referring to rewards in environments of medium and difficult degrees, which reached convergence, respectively, around steps 2000 and 2900.

Figure 8 exemplifies a case in which an agent finds a satisfactory path, that is, after its reward chart has converged at 0.2. In the figure, the path from the lower to the upper region was found by the agent after 2100 steps and represented in 6 frames. Visually, it is noted that the agent did not cause any collision as well as having chosen a short path for the execution of the task, which indicates that the approach and the evaluative analysis on the performance of the task are valid for the context of trajectory simulation of autonomous vehicles.

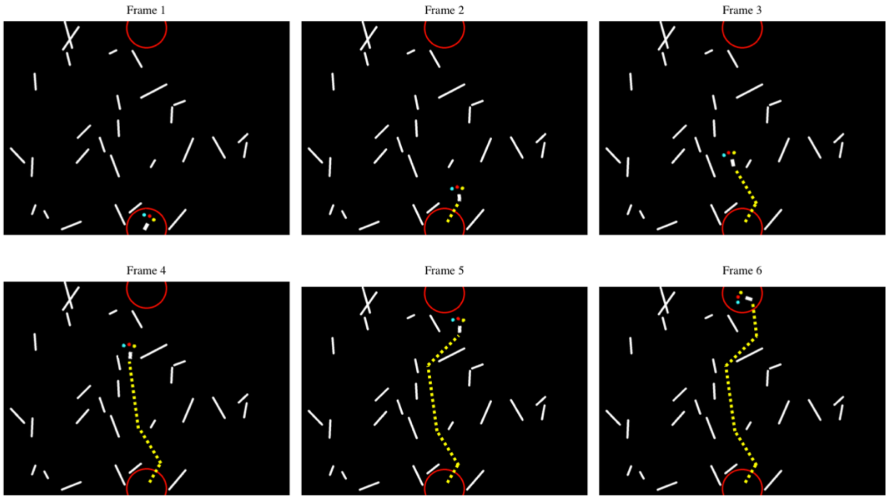

Figure 8: Frames sequence representing the trajectory taken by the agent after the learning has been performed.

3.4 Incremental Evaluation

Also, in the incremental evaluation, 300 testing environments were used in total, divided equally between the 3 degrees of difficulty. For each easy grade environment, an agent was inserted, which after obtaining experience until finding convergence in its rewards graph, this agent was inserted in a medium grade environment until its rewards curve also reached convergence and, finally, each agent was introduced in a difficult environment. At the end of the procedures, the average graphics of rewards obtained by the agents in the environments to where they were inserted with previous learning (that is, medium and difficult environments) were obtained, as presented in Figure 9.

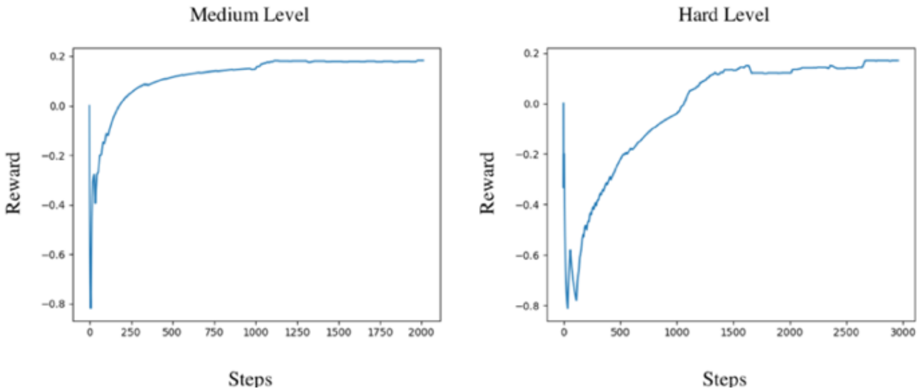

Figure 9: Graphs of average rewards obtained in environments of medium and difficult difficulties by models with prior knowledge.

In comparison with the graphs displayed by Figure 7, it is observed that the graphics in Figure 9 present curves with convergences in fewer steps. In this case, the negative peak regions on the graphic represent the adaptation interactions of the model for the environment where inserted. This region proved to be very small for medium-grade environments. The graphic shows that the agent reaches a positive convergence after approximately 300 steps, still increases the level of convergence around step 1000. What reduces half of the necessary interactions in the environment until confirmed learning compared to the model without prior knowledge. The same is true for the performance of agents in difficult environments, where the number of steps required until positive convergence was around 1200, less than half of the result obtained by the isolated evaluation, which was about 2900 steps, it means, a reduction of upon 58.62% in the number of steps to convergence. This behavior proves the adaptive capacity the proposed model has. It means that actions taken despite adverse conditions are quickly adjusted. Therefore, the model is proved robust to disturbances in the environment.

The validation of the importance of this work was made from a comparative analysis with the performance of related works. Additionally, Table 5 presents the results obtained by the works that most resemble the experiments conducted by this one. Hence, the results obtained by [21] which consist in the application of Deep Reinforcement Learning in the planning of AUVs movement with an evaluation methodology very close to that presented in this work: simulated environments were created for testing in different difficulty levels and the average reward curves are analyzed according to the steps. The results obtained in [22] are also presented, which also proposes the application of Deep Reinforcement Learning for route planning for land vehicles. The main difference between the evaluation methods of the models presented in [22] and in this model is the fact that the reference does not present evaluations of the models due to the degree of difficulty of the tested environments, however, the reward curves are analyzed according to the steps, as in this work, which allows performance comparisons to be made. For comparison purposes, the metric adopted to assess each model is the number of average steps required before the convergence of their respective average reward curves.

Table 5: Comparison between obtained results and related works.

Model	Method	Level	Steps to Convergence
[37]	DDPG	Easy	~2,500
		Medium	~2,800
		Hard	~2,900
	MEP-DDPG	Easy	~2,750
		Medium	~2,900
		Hard	~3,000

[38]	DDPG	Unique	~3,600
Proposed	MLP without previous learning	Easy	~1,500
		Medium	~2,000
		Hard	~2,900
	MLP without previous learning	Medium	~300
		Hard	~1,900

From Table 5 it is observed that the results obtained by the experiments conducted presented results that were competitive with the state of the art. The method using MLP without prior learning surpassed all the results of the related works concerning the Easy and Medium levels and had a similar performance, for the Hard level, to the DDPG method presented by [21]. The proposed model using prior knowledge demonstrated a very large evolution about the results presented by the other models. This means that the model has a fast adaptive capacity even when inserted in a higher-level environment. Thus, the models showed an average reduction in the number of steps until the convergence of 2,550 and 2,500 for the medium and difficult levels respectively (the model presented in [22] was considered a difficult level for this analysis). So, there was a reduction of 89.47% regarding the performance in the medium level and of 64.29% about the difficult level when compared with the related works. Therefore, this approach to path planning through trajectory simulation using an MLP-based policy with prior knowledge has proven to be state of the art for the present task.

4. Conclusion

Two Deep Reinforcement Learning approaches were presented for the path planning task through trajectory simulation. The first approach was about an intelligent agent with a policy based on MLP neural networks that was inserted in environments of different degrees of difficulty without any prior knowledge. The second model went to the first environment, however with previous data in lower degrees of difficulty. The first model obtained competitive results, surpassing the performance of related jobs at the easy and medium levels and conferring similar performance at the difficult level. The second approach, on the other hand, presented a great advance for research in this segment, proving the adaptive capacity of MLP-based models. The results presented show an average reduction in the number of steps up to 76.88% about related works. It is suggested, for future work, the application of the proposed model in an embedded system to a prototype to perform tests in a real environment. Moreover, there must be a concern with the hardware performance, since the system response must be in real-time, thereby suggesting to implement in hardware to optimize the system performance time.

References

[1] S.G. Tzafestas. "Mobile Robot Control and Navigation: A Global Overview". J Intell Robot Syst. 91,35–58, 2018. [Online] Availabe: https://doi.org/10.1007/s10846-018-0805-9

[2] E. Prassler, M.E. Munich, P. Pirjanian, K. Kosuge. Domestic Robotics. In: B. Siciliano, O. Khatib (eds) Springer Handbook of Robotics. Springer Handbooks. Springer, Cham. 2016. [Online] Available: https://doi.org/10.1007/978-3-319-32552-1_65

[3] J. Duperret and D. E. Koditschek. Technical Report on: Towards Reactive Control of Simplified Legged Robotics Maneuvers. October 2017.

[4] Y. Rasekhipour, A. Khajepour, S. Chen and B. Litkouhi. A Potential Field- Based Model Predictive Path-Planning Controller for Autonomous Road Vehicles. In IEEE Transactions on Intelligent Transportation Systems, vol. 18, no. 5, pp. 1255-1267, May 2017.

[5] L. Pan and X. Wang. "Variable pitch control on direct-driven PMSG for offshore wind turbine using Repetitive-TS fuzzy PID control". In Renewable Energy. 2020. [Online] Available: https://doi.org/10.1016/j.renene.2020.05.093.

[6] L. Silveira, F. Guth, P. Drews-Jr, P. Ballester, M. Machado, F. Codevilla, N. Duarte-Filho, S. Botelho. "An Open-source Bio-inspired Solution to Underwater SLAM". In "IFAC-PapersOnLine". Vol. 48, Issue 2, pp. 212-217, 2015. [Online] Available: https://doi.org/10.1016/j.ifacol.2015.06.035

[7] J. Theunissen, H. Xu, R. Y. Zhong and X. Xu. "Smart AGV System for Manufacturing Shopfloor in the Context of Industry 4.0".25th International Conference on Mechatronics and Machine Vision in Practice (M2VIP), pp. 1-6, Stuttgart, 2018.

[8] S.A. Bagloee, M. Tavana, M. Asadi et al. "Autonomous vehicles: challenges, opportunities, and future implications for transportation policies". In J. Mod. Transport. Vol. 24, pp. 284–303. 2016. [Online] Available: https://doi.org/10.1007/s40534-016-0117-3

[9] K. Berntorp. "Path planning and integrated collision avoidance for autonomous vehicles". In 2017 American Control Conference (ACC), Seattle, WA, pp. 4023-4028. 2017.

[10] A. Koubaa et al. "Design and Evaluation of Intelligent Global Path Planning Algorithms". In: Robot Path Planning and Cooperation. Studies in Computational Intelligence. vol 772. Springer, Cham, 2018. [Online] Available: https://doi.org/10.1007/978-3-319-77042-0_3

[11] J. Funke, M. Brown, S. M. Erlien and J. C. Gerdes. "Collision Avoidance and Stabilization for Autonomous Vehicles in Emergency Scenarios". In IEEE Transactions on Control Systems Technology. vol. 25, no. 4, pp. 1204-1216, July 2017.

[12] M. Brown, J. Funke, S. Erlien, J.C. Gerdes. "Safe driving envelopes for path tracking in autonomous vehicles". In Control Engineering Practice. Vol. 61, pp. 307-316. 2017. [Online] Available: https://doi.org/10.1016/j.conengprac.2016.04.013

[13] D. Paley and A. Wolek. "Mobile Sensor Networks and Control: Adaptive Sampling of Spatiotemporal Processes". In Annual Review of Control, Robotics, and Autonomous Systems. Vol. 3, pp 1.1-1.24, May 2020. [Online] Available: https://doi.org/10.1146/annurev-control-073119-090634

[14] C. Cadena et al. "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age". In IEEE Transactions on Robotics, vol. 32, no. 6, pp. 1309-1332, Dec. 2016.

[Online] Available: https://doi.org/10.1109/TRO.2016.2624754

[15] M. Labbé and F. Michaud. RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and longterm online operation. Journal of Field Robotics,pp. 1-31, 2018. [Online] Available: https://doi.org/10.1002/rob.21831

[16] K. Ogata. Engenharia de Controle Moderno. Rio de Janeiro Prentice/Hall do Brasil, 2a Edição, 1993.

[17] H. Oliveira. "Controlo de locomoção do veículo robótico submarino TURTLE com recurso a sistema de variação de flutuabilidade". M.Sc. dissertation. Instituto Politécnico do Porto, Porto, 2017.

[18] A. Dantas et al. "PID Control for Electric Vehicles Subject to Control and Speed Signal Constraints". In Journal of Control Science and Engineering. 2018. [Online] Available: https://doi.org/10.1155/2018/6259049

[19] L. Nie, J. Guan, C. Lu et al. "Longitudinal Speed Control of Autonomous Vehicle Based on a Self-adaptive PID of Radial Basis Function Neural Network".In Iet Intelligent Transport Systems, 2018.

[20] O.C. Arellano, N.H. Romero, J.C.S.T. Mora and J.M. Marín. "Algoritmo genético aplicado a la sintonización de un controlador PID para un sistema acoplado de tanques". In ICBI, vol. 5, n. 10, 2018.

[21] Z. Hu et al. "Deep Reinforcement Learning Approach with Multiple Experience Pools for UAV's Autonomous Motion Planning in Complex Unknown Environments". In Sensors. Vol. 20, 1890. 2020.

[22] L. Yu et al. "Intelligent Land-Vehicle Model Transfer Trajectory Planning Method Based on Deep Reinforcement Learning". In Sensors. Vol. 18, 2905. 2018

[23] R.S. Sutton and A.G. Barto. Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA, 1998.

[24] T. Hester et al. "Deep Q-learning From Demonstrations". In AAAI Publications,Thirty-Second AAAI Conference on Artificial Intelligence, pp. 3223-3230, 2018.

[25] C.J.C.H. Watkins and P. Dayan. "Q-learning".Mach Learn. Vol. 8, pp. 279–292, 1992. [Online] Available: https://doi.org/10.1007/BF00992698

[26] R. Bellman. "The theory of dynamic programming". In RAND Corporation, Proc. National Academy of Sciences, pp. 503–715, 1952.

[27] M.L. Puterman. "Markov Decision Processes—Discrete Stochastic Dynamic Programming". John Wiley & Sons, Inc., New York, NY, 1994.

[28] L. J. Lin. "Reinforcement Learning for Robotics Using Neural Networks". Ph.D thesis, Carnegie Mellon University, Pittsburgh, PA, 1993.

[29] M. Grzes and D. Kudenko. "Theoretical and Empirical Analysis of Reward Shaping in Reinforcement Learning". In 2009 International Conference on Machine Learning and Applications, IEEE, Miami Beach, FL, pp. 337- 344, 2009.

[30] A.P. Braga, A. Carvalho and T. Ludermir. Redes Neurais Artificiais: Teorias e aplicações, 2nd ed. BR: LTC. 2016.

[31] A. Blum. "On-Line Algorithms in Machine Learning". In: Fiat A., Woeginger G.J. (eds) Online Algorithms. Lecture Notes in Computer Science. Vol 1442. Springer, Berlin, Heidelberg, 1998.

[32] S. Ishii, W. Yoshida, J. Yoshimoto. "Control of exploitation-exploration metaparameter in reinforcement learning". Neural Networks. 15(4-6), pp. 665–687, 2002.

[33] D. Ron et al. "On the learnability and usage of acyclic probabilistic finite automata". In 8th Annual

Conference on Computational Learning Theory. pp. 31-40. ACM Press, New York, NY, 1995.

[34] H. Brink, J. Richards and M. Fetherolf, Real-World Machine Learning. 1st ed. USA: Manning Publications, 2016.

[35] T. Masters. Advanced Algorithms for Neural Network: A C++ Sourcebook. 1st ed. New York, NY, USA: Wiley, 1995.

[36] K.Q. Weinberger and L.K. Saul. "Fast solvers and efficient implementations for distance metric learning". In Proceedings of the 25th international conference on Machine learning (ICML ’08). Association for Computing Machinery, New York, NY, USA, pp. 1160–1167. 2008.[Online] Available: https://doi.org/ 10.1145/1390156.1390302

[37] P. Ramachandran, B. Zoph, and Q. V. Le. "Searching for activation functions". arXiv preprint arXiv:1710.05941, 2017.

[38] P. Vamplew, R. Dazeley, A. Berry et al. "Empirical evaluation methods for multiobjective reinforcement learning algorithms". Mach Learn. 84, pp. 51–80, 2011. [Online] Available: https://doi.org/10.1007/s10994-010- 5232-5