Comparison of Reinforcement Learning Algorithms for Motion Control of an Autonomous Robot in Gazebo Simulator

2021 International Conference on Information Technology and Nanotechnology (ITNT) | 978-1-6654-3217-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/ITNT52450.2021.9649145

Daniil Kozlov
Samara National Research University
Samara, Russia
djoade100@gmail.com

***Abstract*—This article compares various implementations of deep Q learning as it is one of the most efficient reinforcement learning algorithms for discrete action space systems. The efficiency of the implementations for the classical Cartpole problem ported to the Gazebo environment is investigated. Then, these algorithms are compared for a self-created bipedal robot problem. Since the creation and configuration of a real robotic system is a laborious process, the initial debugging of the robot can be performed using the appropriate software that simulates the real environment. In our case, the Gazebo simulator was used. Using the simulator allows you to conduct research without having a real robotic system. In this case, it is possible to transfer the results from the simulator to the real system. The result of the study is the conclusion about the greatest efficiency of deep Q-learning with the experience reproduction mechanism. Also, the conclusion is that even for a robot with two degrees of freedom, Q-learning algorithms are not effective enough, and a comparative study with other families of reinforcement learning algorithms is needed.**

Keywords— Reinforcement Learning, Robotics, Simulation, ROS, Gazebo, Gym, Q-Learning, DQN

I. Introduction

Advances in machine learning have brought the ability of computer systems to classify images or recognize objects to approach and sometimes surpass those of humans. Today machine learning methods provide many tools for solving various problems of robotics, such as autonomous motion control, navigation. Finding solutions to these problems can be applied in many practical areas. [1,2] For example, reinforcement learning is the main machine learning method used in robotics.

Reinforcement Learning (RL) is applied in a variety of areas, for example, robotics, finance, healthcare, intelligent transport systems. In general, applications can be grouped into three large groups: automatic machines (autonomous vehicles, smart grids, robotics, etc.), process optimization (scheduled maintenance, supply chains, process planning), and control (for example, failure detection and quality control). [3]

At first, the RL was applied only to simple problems, but the deep RL opened the way to problems of a completely different level of complexity. Deep RL is currently showing very promising results. Unfortunately, many of these breakthroughs are limited to research applications or games, and it is often not easy to bridge the gap between purely research applications and industrial applications. But now they are considering using deep learning to control self-driving cars [4], for trading [5], and energy management for a hybrid electric vehicle [6].

RL in industrial robotics is an area of extremely active research, as it is a natural implementation of this paradigm in practice. The potential and benefits of intelligent industrial robots are vast and varied.

II. Reinforcement Learning

Reinforcement Learning - the part of machine learning that studies sequential decision making to achieve a set goal. The RL task consists of a decision-making agent and the physical or virtual world with which the agent interacts - the environment. The interaction of an agent with the environment is reduced to actions that have some consequences. As a result, the agent receives feedback from the environment in the form of a new state and reward.

Reinforcement learning involves an agent, a set of states S, and a set A of actions per each state. By acting a∈A, the agent transits from state to state. Executing an action in a specific state provides the agent with a reward (a numerical score).

The goal of an agent is to maximize the full reward received over its lifetime. Let us introduce the notation: if a_t - action at time t, and r_t - is the reward at time t, then the agent takes actions a_0, a_1,…,a_t, seeking to maximize the sum:

$$\sum_{i=0}^{t} r_i \tag{1}$$

A. Q-Learning

After Δt steps into the future, the agent will decide the next step. The weight for this step is calculated as $\gamma^{\Delta t}$, where γ (the discount factor) is a number between 0 and 1 (($0 \leqslant \gamma \leqslant 1$) and has the effect of valuing rewards received earlier higher than those received later (reflecting the value of a "good start").

The algorithm, therefore, has a function that calculates the quality of a state–action combination:

$$Q{:}S\times A\rightarrow R \tag{2}$$

Before learning begins, Q is initialized to a possibly arbitrary fixed value. Then, at each time t the agent selects an action a_t, observes a reward r_t, enters a new state s_(t+1) (that may depend on both the previous state s_t and the selected action), and Q is updated. The core of the algorithm is a Bellman equation [7] as a simple value iteration update, using the weighted average of the old value and the new information:

$$Q_\pi(s,a) = E_\pi[r_t + \gamma Q_\pi(s_{t+1},a_{t+1}) | s_t = s, a_t = a] \tag{3}$$

where π – optimal policy, E[·] is the expected value, γ – discount factor.

An episode of the algorithm ends when state $s_{(t+1)}$ is a final or terminal state. However, Q-learning can also learn in non-episodic tasks. If the discount factor is lower than 1, the action values are finite even if the problem can contain infinite loops.

Deep Q-learning used a neural network. Reinforcement learning is unstable or divergent when a neural network is used to represent Q[8].

III. Research tools

The experiment involved the use of the gym framework, for which, in our case, the gazebo acts as the environment. The agent in our case is a simulation of a robot running ROS. Let us take a closer look at these tools:

- ROS [9] (robot operating system) – is a robot programming ecosystem that provides functionality for distributed work. ROS provides standard operating system services such as hardware abstraction, low-level device control, implementation of commonly used functions, interprocess message passing, and package management. ROS has two main "sides": the operating system side ros, as described above, and ros-pkg, a set of user-supported packages (organized into sets called a stack) that implement various robotics functions: SLAM, planning, perception, simulation, etc.
- Gym [10] is a set of tools for developing and comparing reinforcement learning algorithms. The gym library is a collection of test cases - environments. These environments share a common interface that allows you to write common algorithms.
- Gazebo [11] offers the ability to simulate robot populations accurately and efficiently in complex indoor and outdoor environments. You have at your fingertips a reliable physics engine, high-quality graphics, and user-friendly software and graphical interfaces. This simulator is supposed to be used in conjunction with ROS, so the creation of real robots based on virtual ones is the most transparent process, focused on the hardware side. Gazebo is a leader in robot simulation. The purpose of the simulator is to validate and test algorithms, design robots, and train AI using realistic scenarios. This product has the following advantages: high performance physics simulation, realistic environment rendering, generation of data from sensors and sensors, optionally with noise, the ability to use various plugins.
- Unfortunately, even though the gym gives an opportunity for robot training, it does not provide a ROS-based robot training environment using gazebo simulation. Therefore, the openai_ros [12] package was used for the environment. This package is intended to transfer the basic gym concepts and principles for use in conjunction with ROS and Gazebo.All standard paper components have been specified for three reasons: (1)

IV. Cartpole problem

Cartpole - known also as an Inverted Pendulum is a pendulum with a center of gravity above its pivot point. It is unstable but can be controlled by moving the pivot point under the center of mass. The goal is to keep the cartpole balanced by applying appropriate forces to a pivot point.

The rod is fixed in the middle of the bogie moving along the track without friction. A +1 or -1 force is applied to the cart. In the initial position, the pendulum is vertical. The challenge is to keep it upright as long as possible. The episode will end if the pendulum deflects more than 15 degrees or the cart moves more than 2.4 units to the left or right.

The main performance criterion we are considering is the average reward for 100 consecutive tests.

For comparison, we selected the most successful algorithm implementations proposed by users in the framework of competition on the OpenAI platform for the gym environment. These algorithms have been extended to be usable in a Gazebo environment. At the next stage, the effectiveness of these algorithms was compared already in the gazebo environment.

In the original environment, gym CartPole defines "solving" as getting an average award is 195.0 over 100 consecutive trials. Thus, four algorithms of the users n1try [13], mbalunovic [14], rupiexotog [15] were taken. These algorithms showed the results shown in Table 1.

TABLE I. Results of algorithms in gym environment

ALGORITHM	EPISODES BEFORE SOLVE
n1try's algorithm	85
mbalunovic's algorithm	306
rupiexotog's algorithm	933

A. *n1try's algorithm*

This algorithm is the most efficient among those considered, as shown in Table 1.

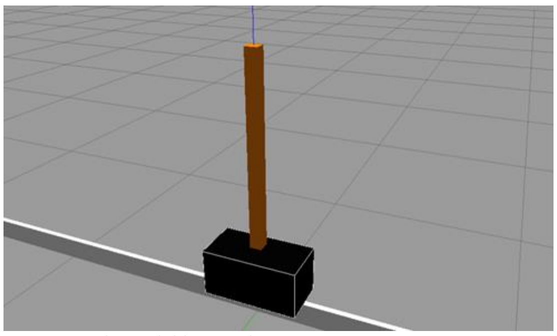
Fig. 1. Cartpole model in Gazebo

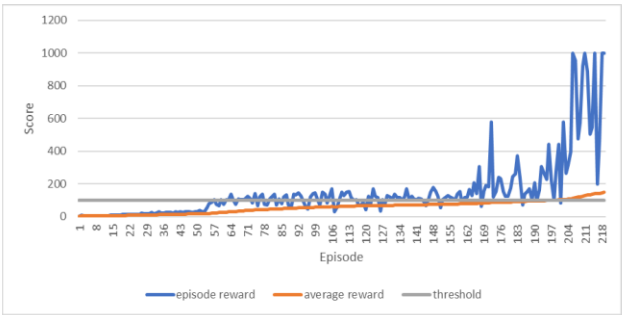

Fig. 2. The results of the implementation of the DQN algorithm by n1try in the gazebo environment. The vertical axis of the graph shows the value of the reward, the horizontal axis shows the number of the training episode

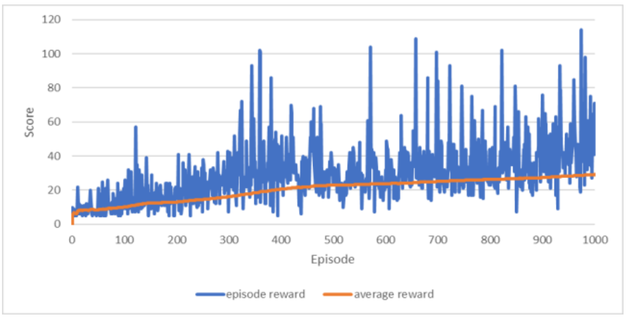

Fig. 3. The results of the implementation of the DQN algorithm by rupiexotog in the gazebo environment. The vertical axis of the graph shows the value of the reward, the horizontal axis shows the number of the training episode

This algorithm is a deep Q learning implementation. This implementation uses a three-layer fully connected neural network.

The algorithm used experience replay, a biologically inspired mechanism that uses a random sample of prior actions instead of the most recent action to proceed. [16]

B. *mbalunovic's algorithm*

This algorithm differs from the previous one in the structure of the neural network used, as well as in the activation functions.

The main difference in the algorithm is the technique used to reproduce the experience. In this algorithm, only the last state of each episode is memorized, and in the past algorithm, all passed states are memorized.

It can be assumed that a different memory mechanism gives a significant deterioration in the result. As you can see from Fig. 3, this algorithm could not solve the problem.

C. *rupiexotog's algorithm*

This algorithm also differs in the structure of the neural network and activation functions. Otherwise, the algorithm repeats the first considered algorithm. This is deep Q learning with an experience replication engine. Similarly to the first algorithm, each state is remembered here during training. The main difference of the algorithm is the smaller batch size, gamma (the discount factor), and alpha (the learning rate).

V. THE SIMPLEST TWO-LEGGED ROBOT

The cartpole task is classical, but at the same time somewhat far from real robotic applications. An interesting problem is the anthropomorphic gait of a robot. However, if we consider systems capable of gait like a person, they will have many degrees of freedom. A problem with many degrees of freedom in the context of reinforcement learning is time-consuming and computationally complex. Therefore, we have created a virtual model of a robot capable of the simplest gait, this robot is shown in Fig. 5. In our case, the task was to withstand the maximum amount of time without falling.

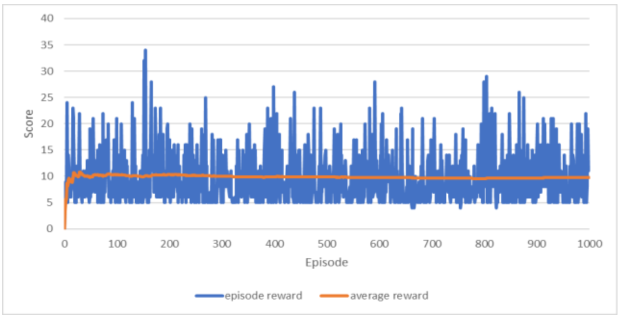

Fig. 4. The results of the implementation of the DQN algorithm by mbalunovic in the gazebo environment. The vertical axis of the graph shows the value of the reward, the horizontal axis shows the number of the training episode

This task is complicated by most of the local minima, which are, in fact, a slow decline. The simplest option is not to take any action at the initial moment, then the robot will roll for a long time until it falls, and such a decision would be better than taking the wrong action. Also, this algorithm strongly depends on the initial actions chosen at random. Individual attempts can come to a standstill from the first actions and do not give any result.

To train the robot, the algorithms described above were taken. In them, the sizes of the input and output layers of the neural network were changed. The number of simulation ticks during which the robot stands above half of its height is set as a reward function. As an observation, the angles to which the legs are bent, and the position and rotation of the robot's body were transmitted.

The learning outcomes are shown in Fig. 6.

As can be seen from the graphs, this task is more complicated for the presented algorithms. In addition to the shortcomings of the algorithm itself, the reasons for ineffective learning can be mistakes made in the formulation of the problem the synthesis of the reward function, an insufficient set of observation parameters, and an incorrectly selected action space.

But the most promising results are still provided by the

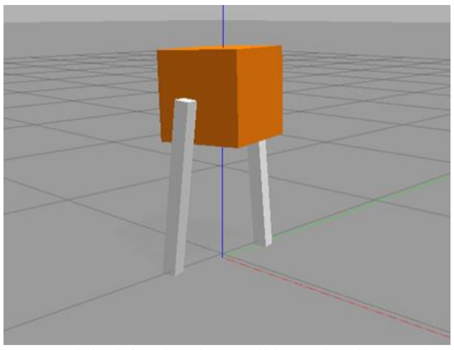

Fig. 5. Our model of the simplest walking robot in Gazebo

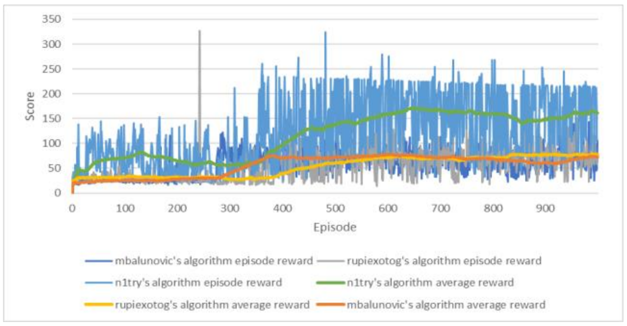

Fig. 6. : Comparison of the results of the work of all the considered deep Q learning algorithms in the Gazebo environment for the simplest two-legged robot. The vertical axis of the graph shows the value of the reward, the horizontal axis shows the number of the training episode

n1try algorithm. The charts show that the algorithm is sensitive to local minima. The n1try algorithm performs best, even though the problem has not been solved. Clearly visible peaks of the graph at the same height can be a consequence of the algorithm reaching a local minimum, reaching a plateau, hitting a saddle point, or reaching a global minimum.

VI. Conclusion

As a result of this study, it can be concluded that reinforcement learning methods are highly dependent on the environment in which the agent is trained. So, the considered algorithms in a gym environment gave acceptable results and guaranteed a solution to the problem. But in the case of the transition to a more accurate simulation of the environment, the earlier working algorithms cease to show themselves well.

Of all the implementations of the Q learning algorithms, the best results are provided by deep Q learning with the experience reproduction technique.

It is worth noting that algorithms were chosen for the study that solved the original problem and was the best in it. Thus, it can be judged that deep Q learning is one of the most promising algorithms in reinforcement learning. In turn, the technique of reproducing experience can improve the quality of training many times over.

But despite this, as it was found out in the experiment with a non-standard model, it becomes clear that it is much more important to correctly configure the reward function for the agent and the conditions under which the episode ends. Since the correct reward function ensures that the agent is only rewarded for useful actions, the correct end-of-episode conditions can shorten training time and make rewards more frequent.

Also, some approaches have the potential for long-term learning. So, the first algorithm, which turned out to be the most successful among the considered ones, is very inclined to a quick search for the local maximum of the reward and cannot leave it for a long time. In turn, the latter algorithm increases the average reward evenly throughout the entire training.

At the same time, it is possible to transfer this experience to a real system with a minimum of modifications in the software. It is only necessary to modify the environment of the robot by adding methods for interacting with the motors into it.

Further research suggests comparing policy gradient algorithms such as NPG, TRPO, PPO, DDPG and their modifications. Consider model-based algorithms such as ME-TRPO. After choosing the most suitable algorithm, a more complex virtual robot model will be synthesized. The new robot model will have greater structural complexity and additional sensors.

Acknowledgment

The work was supported by the Russian Science Foundation grant No. 21-11-00321, https://rscf.ru/en/project/21-11-00321.

References

[1] A. Agafonov, A. Yumaganov, and V. Myasnikov, "Big Data Analysis in a geoinformatic problem of short-term traffic flow forecasting based on a K nearest neighbors method," *Computer Optics*, vol. 42, no. 6, pp. 1101–1111, 2018.

[2] A. Agafonov and V. Myasnikov, "Numerical Route Reservation method in the geoinformatic task of Autonomous Vehicle Routing," *Computer Optics*, vol. 42, no. 5, pp. 912–920, 2018.

[3] R. S. Sutton, F. Bach, and A. G. Barto, Reinforcement learning: An introduction. Massachusetts: MIT Press Ltd, 2018.

[4] B. R. Kiran et al., "Deep Reinforcement Learning for Autonomous Driving: A Survey", IEEE Transactions on Intelligent Transportation Systems, pp. 1–18, 2021.

[5] A. Millea, "Deep reinforcement learning for trading—A critical survey," *Data*, vol. 6, no. 11, p. 119, 2021.

[6] G. Du, Y. Zou, X. Zhang, T. Liu, J. Wu, and D. He, "Deep reinforcement learning based energy management for a hybrid electric vehicle," *Energy*, vol. 201, p. 117591, 2020.

[7] L. Baird, "Residual algorithms: Reinforcement learning with function approximation," Machine Learning Proceedings 1995, pp. 30–37, 1995.

[8] A. Lonza, Reinforcement Learning Algorithms with Python: Learn, understand, and develop smart algorithms for addressing AI challenges. Packt Publishing, 2019.

[9] M. Quigley et al., "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, Jan. 2009, vol. 3.

[10] G. Brockman *et al.*, "OpenAI Gym," *arXiv:1606.01540 [cs]*, Jun. 2016.

[11] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, Sep. 2004, vol. 3, pp. 2149–2154 vol.3. doi: 10.1109/IROS.2004.1389727.

[12] I. Zamora, N. G. Lopez, V. M. Vilches, and A. H. Cordero, "Extending the OpenAI Gym for robotics: a toolkit for reinforcement learning using ROS and Gazebo," *arXiv:1608.05742 [cs]*, Feb. 2017, Accessed: Nov. 24, 2021. [Online]. Available: http://arxiv.org/abs/1608.05742

[13] *OpenAI Gym: n1try's algorithm on CartPole-v0*. Accessed: Nov. 24, 2021. [Online Video]. Available: https://gym.openai.com/evaluations/eval_EIcM1ZBnQW2LBaFN6FY65g

[14] *OpenAI Gym: mbalunovic's algorithm on CartPole-v0*. Accessed: Nov. 24, 2021. [Online Video]. Available: https://gym.openai.com/evaluations/eval_OeUSZwUcR2qSAqMmOE1UIw

[15] *OpenAI Gym: ruippeixotog's algorithm on CartPole-v0*. Accessed: Nov. 24, 2021. [Online Video]. Available: https://gym.openai.com/evaluations/eval_aCiCDmwhTCytFuxMpKoyvQ

[16] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, doi: 10.1038/nature14236.

[17] R. S. Sutton, "Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding," in *Advances in Neural Information Processing Systems*, 1996, vol. 8.

[18] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE*

Transactions on Systems, Man, and Cybernetics, vol. SMC-13, no. 5, pp. 834–846, Sep. 1983, doi: 10.1109/TSMC.1983.6313077.

[19] J. Kober, J. Bagnell, and J. Peters, "Reinforcement Learning in Robotics: A Survey," *The International Journal of Robotics Research*, vol. 32, pp. 1238–1274, Sep. 2013, doi: 10.1177/0278364913495721.

[20] A. Geramifard, C. Dann, R. H. Klein, W. Dabney, and J. P. How, "RLPy: A Value-Function-Based Reinforcement Learning Framework for Education and Research," *Journal of Machine Learning Research*, vol. 16, no. 46, pp. 1573–1578, 2015.