Optimized Deep Reinforcement Learning Approach for Dynamic System

1st Ziya TAN
Erzincan Binali Yıldırım University
Erzincan, Turkey
ziyatan@erzincan.edu.tr

2nd Mehmet KARAKOSE
Firat University
Elazığ,Turkey
mkarakose@gmail.com

***Abstract*— Reinforcement learning methods provide significant and impressive improvements in artificial intelligence studies in recent years, especially in Atari and Go games. This development attracts the attention of scientists who try to understand how people learn. In addition, the biggest advantage of reinforcement learning over other learning algorithms is that it does not require any prior data. This feature distinguishes Reinforcement Learning from others. Reinforcement Learning is an approach in which smart programs work in a certain or uncertain environment to constantly adapt and learn based on scoring. The feedback process is also known as a reward or called a penalty. Given Agents and environment, it is determined which action to take. In this study, we apply the success of Deep Q-Learning (DQL) algorithm, one of the model-free based deep reinforcement learning algorithms used in the literature, on the CartPole problem. We also offer a different method to improve the agent's success during the training phase. We present an optimized DQL algorithm using a function that updates the weights of the neural network at every step.**

Keywords*— *CartPole, Deep Reinforcement Learning, Deep Q-Learning

I. INTRODUCTION

In researches, the human brain is our most complex and mysterious organ, with hundreds of trillion synapses connecting approximately one hundred billion neurons. The digitalization effort of intelligence, which is one of the basic functions of our body, which is called as a combination of abilities such as thought, reasoning, comprehension, effective perception and inference, has prompted scientists to conduct research on artificial intelligence [1].

The term "artificial intelligence", which started to mention itself in the literature in the 1950s, has become an indispensable part of current technologies with its developing structure in the 2020s.

Although DRL assists as an automated learning framework, it might be still time absorbing if we train all controllers from scratch. In the worst case, learning algorithms cannot be able to find the optimal policies for some complex configurations with many degrees of freedom. A common strategy to reduce this problem is to exploit learned policies on simple configurations as prior knowledge that can guide learning for more complex problems [2].

It is a branch of machine learning related to reinforcement learning, decision making and motor control. It examines how an agent can learn how to achieve goals in a complex, uncertain environment, and what they do during this learning. Some features that make reinforcement learning different are as follows:

- Reinforcement learning is a general area as it covers all problems involving decision making [3]. For example; we can give control of a robot's engines to start and jump, make business decisions like pricing and inventory management, or design video games and board games. Sometimes Reinforcement Learning can even be applied to supervised learning problems with sequential or structured output.
- The success of reinforcement learning, especially in DeepMind and AlphaGo games, showed us how good it can be in difficult environments. After these developments, there is an opinion that it can be adapted and used in other environments.

The Q-learning method [4] is a field of reinforcement learning method that is not dependent on any model. In the Q-learning method, the agent provides the ability to learn to behave in the best way in Markovian areas by experiencing the results of the actions, without requiring them to create maps of the fields [5].

The cartpole problem is a famous problem in dynamics and control theory. This naturally creates an unstable system and the pendulum will typically remain vertically downward without any force of swing or dynamic control being applied. The purpose of most cartpole systems is to effectively stabilize the pendulum upward by applying various forces to the pivot point and the axis of motion (horizontal direction).

Beginning to produce something about artificial intelligence, the first thing that comes to mind is to observe human learning in nature. We are interacting with the environment while learning to do something in our daily lives. Therefore, what our behavior will be is affected, and eventually an action occurs. These actions will give us experience in our future interactions [6].

The basic philosophy of reinforcement learning algorithms is to organize a reward or punishment of an action in a method to ensure that a desired outcome occurs [7]. Fig. 1 shows the interaction of an agent with the environment.

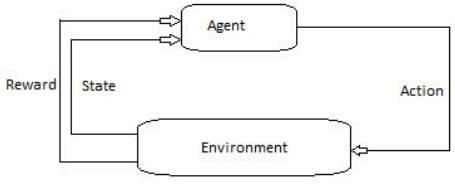

Fig. 1. An RL agent's interaction with the environment [6]

978-1-7281-8603-3/20/$31.00 ©2020 IEEE

In this study we offer a model-free, off-policy Deep Q-learning algorithm. Unlike ordinary DQL algorithm operation, we observed the results by optimizing the weights of artificial neural networks at each step during the training phase.

In this paper, in the second part, Q-Learning algorithm and the operation method of the Deep Q-Learning algorithm are mentioned, and in the third part, the values obtained as a result of the problem definition and solution related to CartPole are explained. In the last part, the results are briefly explained.

II. PROPOSED APPROACH

In this section, the reinforcement learning methods used in our study and the proposed approach will be explained.

A. Q-Learning

The Q-Learning algorithm is one of the most popular algorithms of reinforcement learning. The main purpose in this algorithm is to update the reward it has received after deciding on the next move, and to maximize this reward and select the next step accordingly [8].

The reward map of the agent and its boundaries are determined by us beforehand and the values to be taken during the learning phase are written on the reward table. The agent will have experience according to the values that reward table(q-table) will receive. The agent uses the maximize method to select the places to go from the experience gained in each cycle. He updates these experiences in the Q-Table [9].

Q-table was initialized with 0 at the beginning of the training. At this stage, the agent will have to act randomly. This situation changes when the agent takes the first reward. When the agent arrives at the position he has won the reward, he knows his previous position and writes the value of this position to the Q-table. Table 1 describes the terms used in Q-Learning.The initial Q-learning function can be represented as equation (1) [7].

$$Q(s,a) = r + \gamma maxQ'(s',a') \quad (1)$$

TABLE 1. NOTATIONS

Notation	Meaning
NewQ(s,a)	New Value
Q(s,a)	Current Q Value
α	Learning Rate
R(s,a)	Reward For Taking That Action At That State
γ =	Discount Rate
Max Q'(s',a')	Maximum Expected Future Reward
State (s)	The Parameter That Defines The Current Region
a	Current Action
a'	Next Action
s'	Next State

The flow chart of the Q-Learning algorithm is shown in fig. 2.

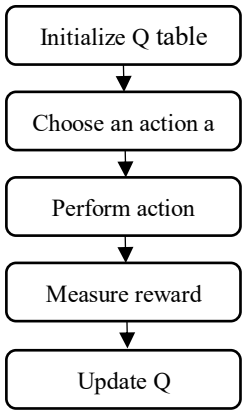

Fig. 2. Flow chart of Q-Learning algorithm

The agent updates the variables of the model every iteration. The pseudo code of the Q-Learning algorithm used at this stage is shown in table 2 [10].

TABLE 2. THE PSEUDO CODE OF THE Q-LEARNING ALGORITHM

1	Initialize Q-values (Q(s,a)) arbitrarily for all state-action pairs
2	For life or until learning is stopped
3	Choose an action (a) in the current World state(s) based on current Q-value estimates (Q(s,.).
4	Take the action (a) and observe the outcome state (s') and reward (r).
5	Update Q(s,a)

B. Deep Q-Learning

In the Q-Learning algorithm, the size of the Q-table depends on the action and state values [11]. Especially if the number of states is excessively high, Q-learning cannot be applied. because Q-table cannot be created. Environments with a high number of states experience problems especially in terms of hardware. Another problem is the state differences between education environment and test environment. The agent trained in the training phase may be unstable and not find solutions when faced with different state in the test phase. It is unclear what the agent will do for a new state in the Q-table [12]. Therefore, Q-learning is not preferred for such problems. For example, in a computer game, since each pixel will be evaluated as state, it will be insufficient in directing the Q-table agent.

As shown in Fig 3, Deep Q-learning takes state value as input and outputs as much as action value.

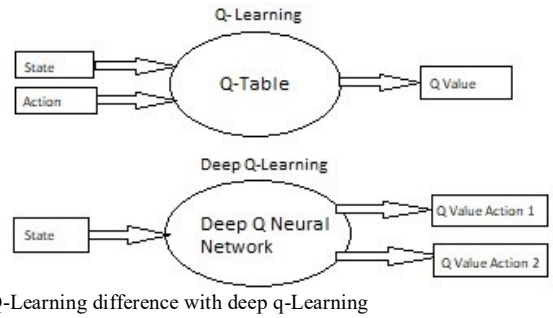

Fig.3. Q-Learning difference with deep q-Learning

While calculating Q-value in Q-Learning, data is taken from the Q-table. In Deep Q-Learning, neural network is used to calculate Q-value approach [13].

The pseudo code of the Deep Q-Learning algorithm is shown in table 3.

TABLE 3. PSEUDO CODE OF DEEP Q-LEARNING ALGORITHM

1	Initialize Replay Memory capacity
2	Initialize network with random weight
3	for each episode:
	-Initialize the starting state
	-For each time step
	-Select an action
	-Via exploration or Exploitation
	-Execute selected action in an emulator
	-Observe reward and next state
	-Store experience and replay memory
	-Update network weights

The flow chart of the Deep Q-learning algorithm is shown in fig. 4.

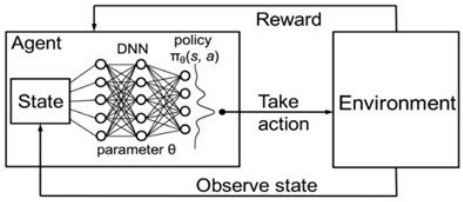

Fig. 4. The Deep Q-Learning algorithm process

For this project, we use CartPole-v0 in OpenAI's Gym package to build our Reinforcement Learning environment.

C. Update Network Weights

The method used to update the target network's weights in the original Deep Q-Network paper [14] is to set them equal to the weights of the network every fixed number of iterations.

D. Experience Replay

When Fig 4 is examined, it is seen that replay memory is initialized in the first stage. At this stage, it stores the state, next state, reward and action values experienced by environment as much as we previously determined. The main goal is to reduce the correlation between the steps. In some cases, the Agent will use it when he wants to take advantage of his past experience. If the correlation does not decrease, the learning of our agent will be negatively affected.

E. Exploration or Exploitation

When the agent explores, he researches extensively and discovers new states. When Agent Exploitation does, he chooses the old experience. The parameter that determines the selection at this stage is the Ɛ-Greedy parameter.

F. $\mathcal{E}$-Greedy

Behavior policy is chosen as Ɛ-Greedy. The Ɛ-Greedy policy allows the agent to choose an action with a fixed value every iteration.

The Ɛ value selected at the beginning is reduced by a certain rate in each iteration, so that the agent learns the environment better. Since the agent will discover the environment due to the large Ɛ value in the first iterations, he will have experience in the next iterations.

$$a = \begin{cases} a, 1-\varepsilon \\ random\ action, \varepsilon \end{cases} \quad (2)$$

The Ɛ in (2) indicates the probability of exploration of the agent in the learning phase.

III. SIMULATIONAL RESULTS

A. Reinforcement Learning for the CartPole System

Gym is a toolkit provided by OpenAI and offered to developers as the Python library for use in their applications. GYM is the collection where multiple environments come together [15].

This section describes which parameters are used to understand how CartPole game is trained in Deep Q-learning algorithms.

As shown in Fig. 7, the system consists of a pole positioned on a carte that moves along the frictionless path. This is connected to pole, carte by joints. The system is controlled by moving it to the right and left by applying +1 or -1 force to carte. The pole is initially positioned perpendicular to the carte. As shown in Table 1, +1 award is given for each step of the pole at certain angles. The episode ends when the angle of the pole with the carte is more than 15 degrees from the vertical or when the carte moves more than 2.4 units from the center [16]. CartPole schematic drawing is shown in fig. 5.

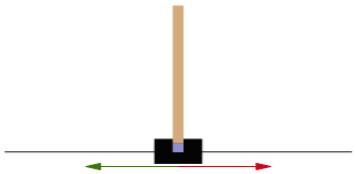

Fig. 5. CartPole schematic drawing

TABLE 4. CARTPOLE OBSERVATION [16]

Num.	Observation	Min	Max
0	Cart Position	-2,4	2,4
1	Cart Velocity	-Inf	Inf
2	Pole Angle	~-41,8°	~41,8°
3	Pole Velocity at Tip	-Inf	Inf

Two different actions in CartPole game are shown in table 5.

TABLE 5. CARTPOLE ACTIONS [16]

Num.	Action
0	Push cart to the left
1	Push cart to the right

B. Training

The python software language was used during the training phase. In addition, keras library [17] was used to create artificial neural networks to be used in deep learning.

The parameters used in the training phase of the Deep Q-Learning algorithm are shown in Table 6.

TABLE 6. PARAMETER OF DEEP Q-LEARNING

Parameter	Value	
Episode	100	
Learning Rate	0,01	
Gamma	0,95	
Exploration Rate	Exploration Max	1
	Exploration Min	0.1
	Decay Rate	0.95

The artificial neural network used in the Deep Q-Learning algorithm consists of 48 single-layer neurons. Mean Squared Error [18] was used as the Loss function. Tanh [19] was preferred as the activation function. Linear function [20] is used as the activation function in the output layer. The training took 45 minutes on a standard equipped computer.

As a result of the training with Deep Q-Learning, the reward-episode graphic in fig. 9 has been formed

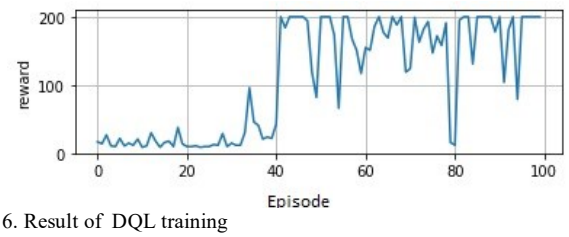

Fig. 6. Result of DQL training

IV. CONCLUSION

Self-learning policy, which is the basic principle in reinforced learning algorithms, has been used in this work at a high level especially in the testing and training stages. Thanks to the dynamic nature of the Ɛ parameter, the decision of the agent is optimized. The agent, whose choice was controlled by the Ɛ-greedy parameter, made his first experiences by exploring the environment. Even in the testing phase, it is provided to be more compatible to the system with its algorithm that can update its experiences.

As a result of the training, the Deep Q-Learning algorithm remained successful in the game for 200 steps, which is the maximum number of steps.

In this paper, we present a successful approach to the CartPole-v0 problem thanks to the weights of the artificial neural network developed to train the RL agent. We started the training and performed the tests on the parameters we determined using the Deep Q-learning algorithm.

According to the data obtained, a successful result was revealed when the parameters and updates in weights of artificial neural network were adjusted appropriately in the Deep Q-learning algorithm.

REFERENCES

[1] Z. Tan, *Derin Öğrenme Yardımıyla Araç Sınıflandırma,* Elazığ: Fırat Üniversitesi-Fen Bilimleri Enstitüsü, 2019.

[2] S. Ha, J. Kim and K. Yamane, "Automated Deep Reinforcement Learning Environment for Hardware of a Modular Legged Robot," *15th International Conference on Ubiquitous Robots,* pp. 348-354, 2018.

[3] V. Tsounis, M. Alge, J. Lee, F. Farshidian and M. Hutter, "DeepGait: Planning and Control of Quadrupedal Gaits Using Deep Reinforcement Learning," *IEEE ROBOTICS AND AUTOMATION LETTERS,* vol. 5, no. 2, pp. 3699-3706, 2020.

[4] C. J. Watkins and P. Dayan, "Q-Learning," *Machine Learning,* vol. 3, no. 8, pp. 279-292, 1992.

[5] L. M. Gambardella and M. Dorigo, "Ant-Q: A Reinforcement Learning approach to the traveling salesman problem," 2000.

[6] Z. WANG, *Knowledge Transfer In Reinforcement Learning: How Agents Should Benefit From Prior Knowledge,* Washington: Washington State University, School of Electrical Engineering and Computer Science, 2019.

[7] R. S. Sutton and A. G. Barto, Reinforcement Learning:An Introduction, London: MIT Press, 2015.

[8] M. Hüttenrauch, A. Šošic and G. Neumann, "Deep Reinforcement Learning for Swarm Systems," *Journal of Machine Learning Research,* pp. 1-31, 2019.

[9] A. Nandy and M. Bisvas, Reinforcement Learning, West Bengal: Library of Congress, 2018.

[10] «Artificial Intelligence,» [Çevrimiçi]. Available: https://www.towardsdatascience.com/. [Erişildi: 2 5 2020].

[11] M. M. Ejaz, T. B. Tang and C.-K. Lu, "Autonomous Visual Navigation using Deep Reinforcement Learning: An Overview," *IEEE Student Conference on Research and Development,* 2019.

[12] T. T. Nguyen, N. D. Nguyen and S. Nahavandi, "Deep Reinforcement Learning for Multiagent Systems: A Review of Challenges,Solutions, and Applications," *IEEE transactions on cybernetics,* 2020.

[13] Y. Liu, D. Zhang and H. B. Gooi, "Optimization Strategy Based on Deep Reinforcement Learning for Home Energy Management," *CSEE Journal of Power and Energy Systems,* pp. 1-10, 2020.

[14] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver and D. Wierstra, "Continuous control with deep reinforcement learning.," *arXiv preprint arXiv:1509.02971,* 2015.

[15] M. Karacabey, A. Qasim, A. Moeed and S. Oehme, "Domain Transfer for Reinforcement Learning Agents," Data Innovation Lab., Münih, 2020.

[16] J. ROY and G. KONIDARIS, "Visual Transfer for Reinforcement Learning via Wasserstein Domain Confusion," *arXiv preprint arXiv:2006.03465,,* 2020.

[17] "Keras-RL Documentation," [Online]. Available: https://keras-rl.readthedocs.io/en/latest/. [Accessed 10 5 2020].

[18] "Mean_squared_error," [Online]. Available: https://en.wikipedia.org/wiki/Mean_squared_error. [Accessed 10 5 2020].

[19] P. Jain, "Complete Guide of Activation Functions," [Online]. Available: https://towardsdatascience.com/complete-guide-of-activation-functions-34076e95d044. [Accessed 10 5 2020].

[20] P. Ramachandran, B. Zoph and Q. V. Le, "Swish: A Self-Gated Activation Function," *arXiv:1710.05941v1 [cs.NE],* 2017.