Tabriz Journal of Electrical Engineering (TJEE), vol. 55, no. 1, 2025 Serial no. 111
DOI: 10.22034/tjee.2024.61074.4824

Employing Chaos Theory for Exploration–Exploitation Balance in Deep Reinforcement Learning

Habib Khodadadi, Vali Derhami*

Department of Computer Engineering, Yazd University, Yazd, Iran.
habib.khodadadi@stu.yazd.ac.ir, Vderhami@yazd.ac.ir
*Corresponding author

Received:03/04/2024, Revised:27/09/2024, Accepted:16/10/2024.

Abstract
Deep reinforcement learning is widely used in machine learning problems and the use of methods to improve its performance is important. Balance between exploration and exploitation is one of the important issues in reinforcement learning and for this purpose, action selection methods that involve exploration such as ε-greedy and Soft-max are used. In these methods, by generating random numbers and evaluating the action-value, an action is selected that can maintain this balance. Over time, with appropriate exploration, it can be expected that the environment becomes better understood and more valuable actions are identified. Chaos, with features such as high sensitivity to initial conditions, non-periodicity, unpredictability, exploration of all possible search space states, and pseudo-random behavior, has many applications. In this paper, numbers generated by chaotic systems are used for the ε-greedy action selection method in deep reinforcement learning to improve the balance between exploration and exploitation; in addition, the impact of using chaos in replay buffer will also be investigated. Experiments conducted in the Lunar Lander environment demonstrate a significant increase in learning speed and higher rewards in this environment.

Keywords
Action selection, chaos theory, deep reinforcement learning, exploration and exploitation.

1. Introduction

Reinforcement learning (RL) is a computational approach for understanding and automating decision-making and goal-based learning, emphasizing learning based solely on direct interaction with the environment without relying on a supervisor or a complete model of the environment [1]. RL has been used in various issues such as robotics, resource allocation, and cloud computing [2]. However, real-world problems have very large state spaces that classical (discrete) reinforcement learning methods are unable to solve. Recent advancements in machine learning have led to the emergence of deep neural networks which are used for automatic feature extraction and other applications. One technique for using reinforcement learning in complex and high-dimensional problems is to combine it with deep neural networks, which is called deep reinforcement learning; one type of these networks, created by combining the Q-learning method with deep neural networks, is called Deep Q-Network (DQN) [2]. The importance of exploring the rewards of various actions cannot be overstated when it comes to optimizing RL algorithms. By actively seeking out and evaluating different outcomes, exploration plays a key role in guiding behavior within the state space, ultimately aiding in the swift convergence towards an optimal policy. Furthermore, exploration is essential in uncovering the underlying reward system of the environment, thus assisting in the identification of the most effective policies. For effective learning in reinforcement learning, actions should be selected in a way that the environment is explored properly and the knowledge acquired during learning is utilized appropriately. Completing these two tasks simultaneously is not possible, and a balance between exploration and exploitation must be created in RL. Various methods have been proposed to create this balance between exploration and exploitation, but this issue has not been fully resolved and research in this area is ongoing [1].
Reinforcement learning exploration can be divided into two primary categories: efficiency and safe exploration. In terms of efficiency, the goal is to increase the effectiveness of exploration to allow the agent to explore with as few steps as necessary. On the other hand, safe exploration prioritizes ensuring the agent's safety during the exploration process. Efficiency-based methods can be divided into two categories imitation-based (Emulate learning from teacher) and self-taught methods (Learning from scratch). There are various categories of self-taught methods, including planning, intrinsic rewards, and random exploration. In planning, the agent strategically determines its next actions to improve its understanding of the environment. On the other hand, in random exploration, the agent does not deliberately plan its actions but instead explores the environment and observes the outcomes of these explorations. In intrinsic reward methods, the agent is rewarded for visiting new states or

new behaviors [3]. Some of the methods available in each of these categories are listed in the related works section. Based on the chaotic features such as sensitivity to initial conditions, pseudo-randomness, ergodicity and non-periodicity, numbers generated by chaotic systems have been used in various applications such as different types of encryptions (image [4-5], audio [6], text [7], and video [8]), evolutionary algorithms [9-11], and prediction [12]. In these applications, various chaotic maps and different methods have been used for generating chaotic numbers for use in each of these cases. Chaos theory has shown positive performance against random data in these applications. For example, chaos has improved and increased the speed of finding the global optimum in most evolutionary algorithms, or many encryption algorithms are based on chaos. This better performance is due to the unique properties of chaotic systems. Research on creating stronger chaotic systems continues, and new systems are being introduced [13].
Chaotic numbers generated for use in action selection methods in reinforcement learning and deep reinforcement learning can also be utilized.
The use of deep reinforcement learning is increasing widely in machine learning problems, and any improvement in its performance is important; Since there is no prior work on the use of chaos in the action selection part of deep reinforcement learning, in this paper, chaos in the action selection part of deep reinforcement learning and replay buffer have been used, leading to an increase in learning speed and earning more rewards in the tested environment. The main contributions of this paper are summarized as follows:

- in order to create a balance between exploration and exploitation, chaos in the action selection part of deep reinforcement learning will be used for the first time (As far as we know) in this paper using a ε-greedy approach. The ergodicity property of chaos theory causes the DQN performance to be significantly improved.
- Using chaos to select samples from the replay buffer in deep reinforcement learning to further diversify the selected samples and increase the speed of network training, which so far has no precedent in this regard.

The structure of this paper is as follows: First, in part 2 and 3, related works and chaotic systems are briefly introduced, followed by an overview of reinforcement learning and deep reinforcement learning in Section 4. The proposed method in this paper is presented in part 5 and its computational results are discussed in Section 6. Finally, discussion and conclusion are presented in parts 7 and 8.

2. Related Works

So far, numerous research studies have been conducted in the field of action selection and improving the balance between exploration and exploitation. In research [14], a method called Stochastic Curiosity Maximizing Exploration (SCME) for incentivizing agents in reinforcement learning to explore their environment more effectively has been developed. This paper introduces a new method called Stochastic Curiosity Maximizing Exploration (SCME) for SCME uses an intrinsic reward signal that encourages the agent to visit novel states and take actions that lead to new experiences. The method achieves good results compared to existing exploration algorithms on a range of tasks and environments.
MIMEx [15], a method for generating intrinsic rewards in reinforcement learning tasks by using masked input modeling is introduced. The authors propose a novel reward signal that encourages the agent to focus on relevant parts of the input by masking out unimportant information. They validate their approach on a range of tasks and show that incorporating MIMEx leads to improved performance compared to traditional reward mechanisms. Overall, MIMEx offers a promising approach for enhancing the learning capabilities of reinforcement learning agents.
The use of adaptive learning rates in fuzzy reinforcement learning is also mentioned as a way to achieve a balance between exploration and exploitation [16], where the learning rate is adjusted considering the "fuzzy visit value" of the current state.
In another study, Cuckoo Search Algorithm were used instead of traditional action selection methods for action selection [17]. In this method, the reinforcement learning problem is shown as an optimization problem where the candidate solutions are the values of Q and the objective function is the Q-function. In each iteration, the Q-learning algorithm updates combinations of Q values and actions using the evolutionary optimization algorithm, and the action with the highest optimized value is selected. The steps and implementation of the idea were also tested on several environments such as MAB and Cliff-Walking, and the effectiveness of this method was confirmed.
In a paper titled "First Return, Then Explore" a new way to explore is presented by suggesting that agents should prioritize returning to areas of high reward before exploring new options [18]. This approach focuses on encouraging agents to first exploit known strategies before exploring new options. This can help improve the efficiency and effectiveness of the learning process, ultimately leading to better performance in reinforcement learning tasks. The primary method utilized in this approach involves remembering visited states and paths, as well as learning from the goal by assuming the goal state is known but the path to it is unknown.
Another study has been attempted to determine the exploration rate through a model of environmental entropy [19]. Entropy-based exploration (EBE) allows an agent to effectively explore unfamiliar areas within the state space. By measuring the agent's progress in a given state using only action values that are specific to that state, EBE dynamically navigates the state space, prioritizing exploration in regions that have not yet been thoroughly investigated.
Safe exploration methods include using human designer knowledge to create boundaries for the agent. For example, in one study, the actions of an agent are limited by an additional pre-trained module to prevent unsafe actions [20]. Additionally, in research, the states of the investigated problem and their advantages are stored in a shared memory, and states that are frequently visited but have low advantage receive extra negative rewards. Here, the agent is allowed to use previous experiences to identify undesirable states [21].

In another research, graph-based exploration method has been used for multi-agent environments [22]. This paper presents a technique for effective multi-agent Q-learning by utilizing graph exploration. The approach involves creating a graph structure to represent the relationships between agents and states in the environment. This graph is then used to guide exploration and improve the learning process. The authors demonstrate the effectiveness of their approach through experiments on various multi-agent scenarios.

The history of using chaos in reinforcement learning to create a balance between exploration and exploitation is not extensive, and in most cases [23-27], logistic map has been used to generate random numbers for use in ε-greedy action selection methods. Experiments on the Target Capturing Task problem have been conducted to compare chaotic and traditional conditions; the chaotic method has shown better results with less time and distance spent to find the target [23-24]. The authors of these papers have also performed this action on the Shortcut Maze problem and achieved similar results [25]; furthermore, in another study, the Tent mapping has been used and found to be less efficient compared to logistics; the experimental environment is the same as the Shortcut Maze environment, which is a type of maze-like environment [26]. In these papers, SARSA and Q-learning algorithms have been used as reinforcement learning methods. Considering that in these studies, the environment may have slight changes, a performance criterion $q(n) = t_{min}/t_n$ has been introduced; where t_{min} is the minimum distance required to reach the target from the beginning and t_n is the actual distance covered, with the best case scenario having a q(n) value of 1. For example, in the Shortcut Maze problem, with 900 executions including movement from the start to finding the target, the average value of q in the case of using random values in the ε-greedy method in 899 runs was reported as 0.86, and using chaotic system values in the logistics method it was reported as 0.91 [25]. Additionally, in another research, action selection in the ε-greedy method was compared to chaotic and traditional ε-greedy methods on the Shortcut Maze problem where logistics, Tent, and Chaotic Neuron mappings were used, and the chaotic methods showed better performance [27].

In a study related to reinforcement learning, the application of chaos theory in dynamic programming to overcome global updates of all states has been examined, utilizing the logistic chaotic system for this purpose [28]. This method involves executing policy evaluation once in each stage of policy iteration and updating only a few states proposed by the chaotic system. The policy improvement stage then follows, using similar procedures in the value iteration method, which resulted in better outputs than the conventional method.

3. Chaotic Systems

Chaotic systems are non-linear dynamic systems that are very sensitive to their initial conditions and exhibit a pseudo-random behavior. A slight change in the initial conditions of such systems will lead to significant changes in the future. The definability of the system and the certainty despite the pseudo-random behavior are also important characteristics of chaotic systems [29].

For a system to be classified as chaotic, it must exhibit the following properties [29]:

- **Sensitivity to initial conditions**: This characteristic of chaotic systems shows that even a small adjustment to the starting conditions can lead to substantially different outcomes over time.
- **Topological mixing or topological transitivity (ergodicity)**: is a characteristic that states chaotic variables will move through all states within a set range without repeating. This property can be utilized as an optimization tool to ensure that no solution is revisited in the search space, preventing algorithms from getting stuck in a local optimum. this feature leads to the generation of diverse and non-repeating numbers.
- **Topological density**: refers to the property that every point within a given space is approached by periodic orbits in an arbitrary manner.

Substituting chaos for random numbers has been shown to increase learning speed due to leveraging the special properties of chaos, especially its ergodicity property [29].

So far, many chaotic systems have been introduced. The logistics system is a chaotic system. The governing equation for this system is in the form of (1) [4]:

$$x_{n+1} = \lambda x_n(1 - x_n) \qquad x_0 \in [0,1] \qquad (1)$$

In which, for values of λ that are in the interval [3.56,4], the system exhibits chaotic behavior. Giving any initial value for a chaotic system, the next numbers are obtained according to the mapping relationship. Of course, it should be noted that even with very slight changes in the same initial value, the generated number series will be completely different from each other.

In Figure 1, the behavior of the logistic system with initial value x0=0.6000 and parameter value λ=3.9999 is shown; also, in Figure 2, the first 50 numbers generated by this system are shown.

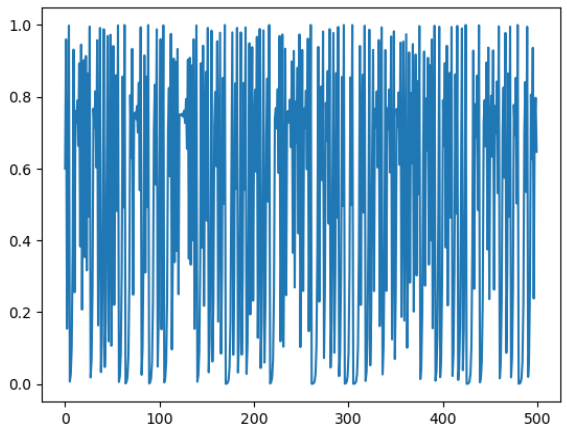
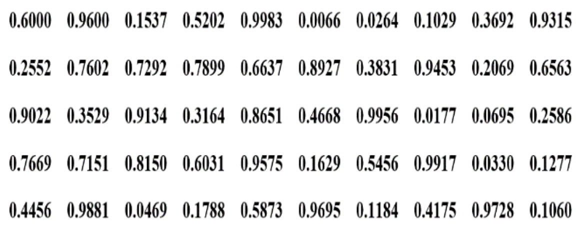


Fig. 1. The chaotic behavior of logistic signal in the first 500 iterations with x0=0.6000 and λ=3.9999 is shown in the plot (horizontal axis shows the number of iterations and vertical axis shows the value of logistic signal in each iteration).

In this figure, another characteristic of chaotic systems is apparent; it can be observed that the generated numbers are well distributed in the space of 0 to 1 and after several iterations, different parts of the space are visited.



0.6000	0.9600	0.1537	0.5202	0.9983	0.0066	0.0264	0.1029	0.3692	0.9315
0.2552	0.7602	0.7292	0.7899	0.6637	0.8927	0.3831	0.9453	0.2069	0.6563
0.9022	0.3529	0.9134	0.3164	0.8651	0.4668	0.9956	0.0177	0.0695	0.2586
0.7669	0.7151	0.8150	0.6031	0.9575	0.1629	0.5456	0.9917	0.0330	0.1277
0.4456	0.9881	0.0469	0.1788	0.5873	0.9695	0.1184	0.4175	0.9728	0.1060

Fig. 2. The first 50 numbers of logistic signal with x0=0.6000 and λ=3.9999.

Another chaotic system is the Zhang system, which is constructed by combining two chaotic logistic and Tent systems, and its relationship can be seen in equation (2) [30].

$$x_{n+1} = 2\mu|x_n|(1-2|x_n|) \qquad (2)$$

In this paper, logistic and Zhang systems are used to generate chaotic numbers, which will be used instead of random numbers in the action selection and replay buffer of deep reinforcement learning. In order to generate chaotic numbers using these systems, every time a random number is given as an initial value to the chaotic system and its output is used; Also, in addition to this method, by giving an initial value to the chaotic system, and using the generated number as the next value for the input of the system, we can have a sequence of chaotic numbers with a specific initial value, which can be reproduced by having the initial value of this sequence. In this paper, the first method was used to generate chaotic numbers, and there is no dependence of chaotic numbers on each other in the form of a sequence.

4. Reinforcement learning and deep reinforcement learning

Reinforcement learning is the process of learning appropriate actions from a set of allowable actions for a specific situation based on received rewards and penalties [1]. The key idea of reinforcement learning is to use value functions to find appropriate policies, and dynamic programming is one of the methods of reinforcement learning that uses the Bellman equation to calculate the value of each state from the environment or the value of state-action pairs (equations 3 and 4) [1]. To calculate the value of each state, the values of other states are used.

$$V^{\pi}(s) = \sum_a \pi(s,a) \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V^{\pi}(s')] \qquad (3)$$

$$Q^{\pi}(s,a) = \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V^{\pi}(s')] \qquad (٤)$$

In these equations, action a is selected from the set of actions in state s, and the next states s' are members of the set of states. $V^{\pi}(s)$ is the value of state s under policy π, $Q^{\pi}(s, a)$ is the value of action a in state s under policy π, $P^a_{ss'}$ and $R^a_{ss'}$ are the transition probability and the expected reward value to the next state, respectively. Also, π (s, a) is the probability of selecting action a in state s and finally, γ is the discount factor.

There are multiple algorithms in reinforcement learning, and one of the most commonly used ones is the Q-learning algorithm. In the Q-learning algorithm, all the values of state-action pairs are stored in a Q-table, where the rows represent states and the columns represent actions, and each element in this table is an estimate of the optimal value of a state-action pair. In each step of the agent's movement in the environment, this table needs to be updated, and new estimated values should replace the previous values using the received rewards. The update rule of this algorithm is given by equation (5) [17].

$$Q(s_t,a_t) = Q(s_t,a_t) + \alpha(r_{t+1} + \gamma \max_a Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)) \qquad (5)$$

In this algorithm, α is the learning rate and r is the instantaneous reward value. Here, the learned state-action value function Q directly approximates the optimal state-action value function independently of the policy that dictates behavior.

Many real-world problems have very large state spaces and it is not possible to maintain a Q-table, in these cases an approximator is used to approximate the state-action value. There are different structures for this purpose, one of which is using artificial neural networks. The combination of deep learning and reinforcement learning has led to the emergence of deep reinforcement learning. The deep Q-network (DQN) is created by combining the Q-learning method with deep neural networks, where a neural network is used to estimate the state-action value [2]. As shown in Figure 3, in deep reinforcement learning, various types of neural networks and related concepts are used to estimate the state-action values due to the large number of states or continuous state inputs, and the output of the network is the estimated state-action values.

Since in machine learning there is a need for independent and identically distributed (IID) samples from the problem space, previously observed samples are stored in memory (Replay buffer) and batches of these samples are randomly selected from this memory for network training each time [2].

Efforts have been made to increase the speed and efficiency of DQN, including the use of Double Q-learning [31] to overcome the problem of overestimation and prioritized experience replay [32] to utilize more important experiences in the learning process.

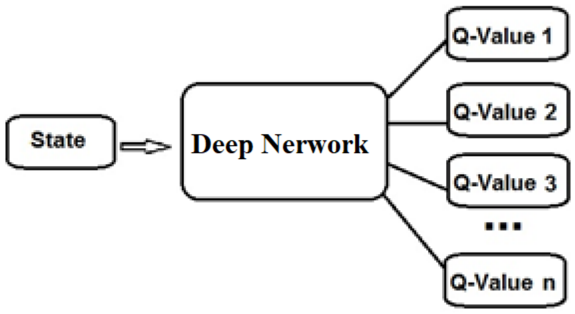


Fig. 3. Deep Q Network: Using a deep network to estimate the values of state-action pairs.

One of the important improvements on DQN is the method Dueling DQN, which as shown in Figure 4 and equation (6), estimates values V and A (advantage) instead of directly estimating Q values by the network, and by combining these values, Q values are estimated, leading to an improvement in network speed. The reason for the increase in network speed in this algorithm is

related to the simplicity of estimating V compared to Q, because instead of estimating state-action values, which is a larger space, it only estimates values and the problem space will be smaller as a result, so the network will reach a conclusion sooner [33]. It is important to note that in many cases, estimating the value of each action is unnecessary; in some cases, knowing which action to take is crucial, but in many other cases, the choice of action has no impact on what happens [33].
In RL, at any given moment, there are two strategies for action selection: one can choose a greedy action, which is the action with the highest estimated value and therefore utilizes the current knowledge; or one can choose a different action, in which case the agent's effort will be focused on further exploration, allowing the agent to improve its estimation of other non-greedy actions. For effective learning, actions should be selected in a way that the environment is adequately explored and penalties are avoided. It is not possible to simultaneously perform these two tasks fully, and a balance between exploration and exploitation must be established. The use of greedy policies in action selection limits the agent to a small part of the environment and hinders the discovery of other parts of the environment and finding better policies; therefore, other action selection methods are used.

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(a) \tag{٦}$$

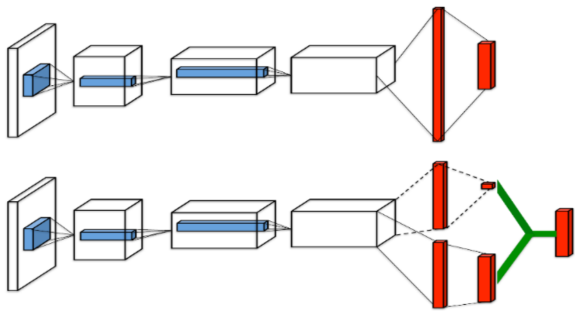


Fig. 4. The upper figure represents the traditional DQN approach, while the lower figure represents the Dueling DQN approach. In the DDQN method, the Q value estimation is obtained from a specific combination of value estimates and advantage values (Equation 6) [33].

In general, solutions for action selection and maintaining balance in reinforcement learning are divided into two categories of direct exploration methods and indirect exploration methods. In direct exploration methods, it is assumed that some information about the environment such as the state transition function and reward function is available, which is usually not the case in reinforcement learning problems; therefore, indirect exploration methods are often used in RL problems [1]. Some indirect exploration methods include random exploration, greedy exploration, optimistic initial values exploration, ε-greedy exploration, soft-max exploration, and upper confidence bound exploration [34].
In the ε -greedy approach, with a probability of 1-ε (where ε is a positive real number between 0 and 1), the action with the highest value is selected, and with a probability of ε, all actions can be chosen (equation 7). The results of applying this method indicate higher efficiency compared to the greedy approach. Due to its low computational overhead, this method has been widely used in many problems [34].
When choosing an action using the ε-greedy method, we need to generate a random number and make a decision based on it. Firstly, by generating a random number and comparing it with epsilon, a decision is made whether to select the action with the maximum value or to choose one of the actions randomly. Secondly, if it is decided based on the first step to choose one of the actions randomly, another action will be selected by generating another random number.

$$P(S,a) = \begin{cases} 1-\varepsilon+\frac{\varepsilon}{N} & a = argmax_{b\in A}\, Q(S,b) \\ \frac{\varepsilon}{N} & otherwise \end{cases} \tag{٧}$$

5. Proposed Method

In the ε-greedy approach, numbers generated by chaos can be used instead of random numbers, utilizing the chaotic property to achieve a better balance between exploration and exploitation; here, instead of generating random numbers and making decisions based on them, numbers generated by the chaotic system are used for decision-making. Based on previous research [23-27], using chaotic systems in the action selection component of RL methods has led to a better balance between exploration and exploitation in maze-like environments. In this paper, we will be using chaos in deep reinforcement learning for the first time. For this purpose, after initial preprocessing and setting appropriate values, numbers generated by chaotic systems replace random numbers in the ε-greedy action selection process, utilizing the chaotic property to achieve a better balance between exploration and exploitation.
In addition to the action selection part, in many deep reinforcement learning algorithms, a part called replay buffer is used to store samples for training the network. Each time, a number of samples are randomly selected from this memory for network training. Here, for selecting samples from the replay buffer, chaos is used as described below. we expect that by utilizing chaos in the replay buffer, a variety of samples will be available for training the network, leading to improvement. For this purpose, first chaotic numbers are generated and then they are mapped to replay buffer samples and these samples are selected for training the network, which was done randomly in the traditional mode.
For example, to select 32 samples from a memory of one million using chaos, first we generate 32 chaotic numbers with the desired chaotic system, where the numbers should be between 0 and 1; then we multiply these numbers by 10^{14} and round them. The remainder of the obtained number divided by one million determines the selected sample. Considering the properties of chaos, we expect to have a more diverse set of samples for training the network, leading to improvement.
To test and implement this idea, the Lunar Lander environment, which is one of the environments available in OpenAi Gym [35], was used. The components of this environment include a spacecraft that needs to land on a

specified surface of a planet (between two yellow flags in image 5); therefore, the farther the landing distance from these two flags, the higher the negative score. The action space of this environment is discrete and has 4 actions, including do nothing (descending straight down as it falls), or activating one of the three engines: up, left, or right, and using them to fly in any direction. However, the state space of this environment is continuous, with an 8-dimensional vector including horizontal and vertical position (-1.5 to 1.5), horizontal and vertical velocity (-5 to 5), angle (-3.14 to 3.14), angular velocity (-5 to 5), and two binary values indicating contact with the ground on the left and right foot. Since the state space of this environment is continuous, deep learning methods and the use of a neural network as an approximator must be used to solve it, rather than tabular-based reinforcement learning methods.

Moving from the top of the screen to the landing pad and successfully coming to rest will earn you approximately 100-140 points. However, moving away from the landing pad will result in a loss of reward. Crashing will incur an additional penalty of -100 points, while successfully coming to rest will yield an extra +100 points. Each leg with ground contact will earn you +10 points. Using the main engine will deduct -0.3 points per frame, while using the side engine will deduct -0.03 points per frame. Finally, reaching a successful landing is worth 200 points.

The closer the landing location is to the designated place, the higher the score, and the farther it is, the lower the score. The further the spaceship deviates, the lower the score. The slower/faster the spaceship moves, the score increases/decreases. Ultimately, if each run earns at least 200 points, it is considered a solution.

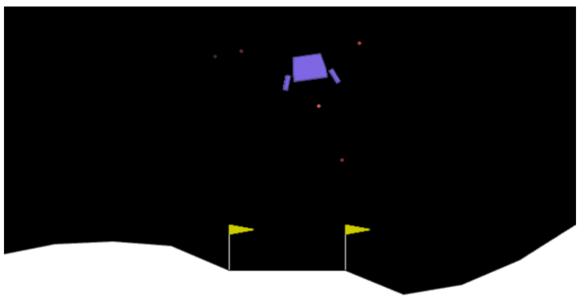


Fig. 5. Lunar Lander environment is one of the environments in OpenAi Gym [35].

6. Experiments results

The Lunar Lander environment and the Dueling DQN algorithm were used in this study to test the use of chaos in the action selection part of deep reinforcement learning and the replay buffer part. Google Colab was used for implementation. In this study, an ε-greedy action selection method was used, with epsilon starting at 1 and gradually decreasing at a rate of 0.95 until reaching 0.1 where it remained constant. In the replay buffer part, instead of randomly selecting samples, the Zhang chaotic system was used to select samples for training the neural network. In the action selection part, a combination of the Zhang system and logistics was used; a number generated by the Zhang system was first obtained, and if it was smaller than the epsilon value, the logistic system was used for chaotic selection, otherwise the maximum action was chosen. To select one of the four operations using the logistic system, we first generate a chaotic number by providing an initial value to the logistic system; then we multiply this number by 10^{14} and round it. The remainder of dividing the obtained number by four determines the desired operation. Since the role of chaos in the study was investigated, all parameters in both methods remained completely constant and only the mentioned changes and use of chaos instead of randomness were applied. The value of μ in the Zhang equation was set to 2.4140 and the value of λ in the logistics equation was set to 4.

In figures 6 to 9, the results of using chaos in this environment are observable. The vertical axis shows the average reward received in the last 50 runs, while the horizontal axis represents the runs. Here, the amount of replay buffer is 1000000, each time, 32 examples of this memory are selected using chaos to train the neural network.

The neural network used is a fully connected feedforward network with 128 neurons in each of the first and second layers. The activation function used is ReLu, the loss function is MSE (Minimum Square Error), and the Adam algorithm is used for updating the network weights. The results of the experiments are also shown in Table **I**; in this table, the average reward received in all 200 runs, the reward received in the 200th run, and the average reward received in the last 50 runs are compared for 4 different methods.

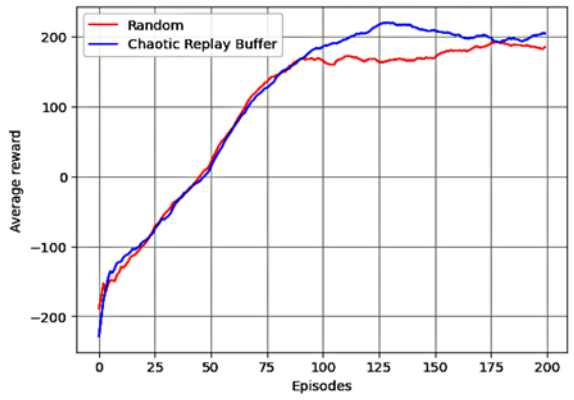


Fig. 6. Comparison of the rewards received in the Lunar Lander environment in the chaotic sample selection mode from the replay buffer with the random sample selection mode.

Based on figures 6 to 9 and Table **I**, the superiority of using chaos over using random mode is clear. When the chaotic ε-greedy method is used instead of random selection of actions in the ε-greedy method, the results are very promising and the average reward received increases rapidly and almost climbs to above 200 in the 65th run, which shows the efficiency of the new method, while in the traditional method, even until the 200th run, the average reward received has not reached 200, and practically the environment has not been learned. The graph of the average reward received in the traditional mode (red), the use of chaos only in the replay buffer (green), the use of chaos only in the part of the action selection (black) and the use of chaos in both parts of the replay buffer and action selection (blue) is shown in Figure 9.

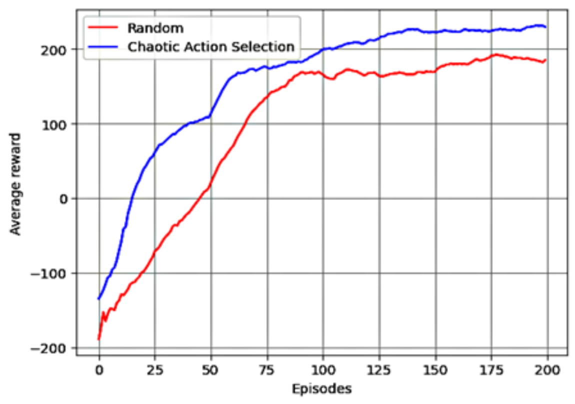


Fig. 7. Comparison of rewards received in the Lunar Lander environment in the chaotic action selection mode versus random action selection mode.

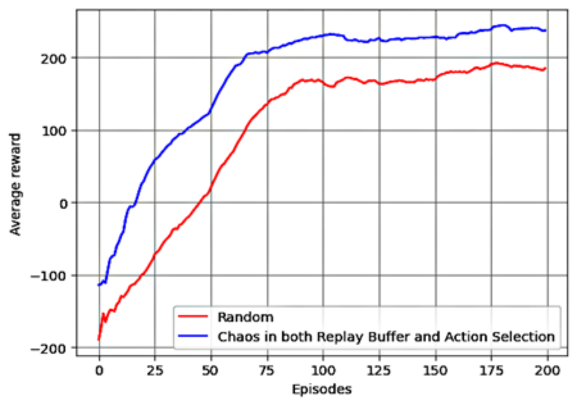


Fig. 8. Comparison of rewards received in the Lunar Lander environment in the mode of simultaneous use of chaos in the replay buffer and action selection.

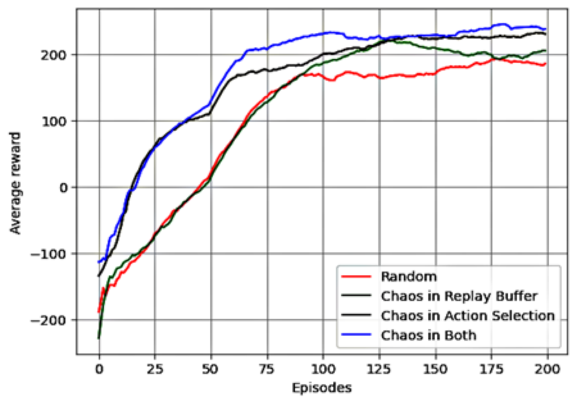


Fig. 9. Comparative chart of the rewards received in the Lunar Lander environment in the chaos mode; comparison of figures 6, 7, and 8.

As seen in these figures; The use of chaos causes faster learning of the problem and the problem converges faster than the normal state, and the effectiveness of chaos has been shown with these experiments.

In addition to the epsilon-greedy method, random and greedy action selection methods were also tested. In the greedy action selection, at each step, the action with the highest value is chosen, while in the random action selection method, one of the actions is chosen randomly each time. The experimental results showed that using these two methods did not lead to learning the environment and did not solve the problem.

Table I. Comparison of rewards received in traditional mode and chaos mode in the Lunar Lander environment.

	Total rewards in the 200th run	Average reward per 200 runs	Average reward over the last 50 runs
Traditional ε-greedy method	181.4	133.7	185.0
Chaos only in the replay buffer	218.8	151.1	204.6
Chaos only in action selection	247.0	166.5	230.1
Chaos in both parts	**273.0**	**204.3**	**237.2**

7. Discussion

It should be noted that the aim of this paper is not to present a new reinforcement learning algorithm, but rather the results obtained are simply the result of replacing random numbers with chaotic numbers and no other parameters have changed. The results of the previous section showed that replacing random numbers with chaos in the action selection and replay buffer significantly improved the performance of the learned algorithm, primarily due to the special characteristics of chaos and most importantly its ergodicity property. Here, no new method has been presented, and there is no claim of superiority over existing established methods. Since one of the conventional action selection methods in deep reinforcement learning is the ε-greedy, which uses random numbers to select one of the actions, chaos was employed to enhance the efficiency of this method, yielding favorable results, and a complete comparison with this method was carried out. In addition to the ε-greedy, which is the basis of this paper, random and greedy action selection methods were also tested. However, these methods did not converge under the conditions described in this paper.

According to these results, although the use of chaos in the action selection is much more effective than the use of chaos in the replay buffer, but the combination of the effects of both of these is better than the individual cases, and therefore, the use of chaos in both parts is recommended. If chaos is used only in the replay buffer, although it is superior to the random mode, it is not as good as using chaos in action selection.

Since the methods of generating random numbers are based on specific algorithms, the possibility of regenerating and generating similar numbers exists, but in chaos, this is not the case considering the properties discussed. Therefore, using chaotic generated numbers in selecting actions in the ε-greedy method is more compatible with the goals of this method and improves its performance as observed in the results.

Since chaotic numbers can be easily generated from equations (1) and (2), generating numbers using logistic and Zhang systems does not impose significant computational complexity. In an experiment, the time to generate 1000 uniformly random numbers was approximately 0.059 seconds, and the time to generate 1000 chaotic numbers using the logistic system was approximately 0.098 seconds. Although the time in the experiments used is not overly significant.

Generating chaotic time series with chaotic maps typically requires more computations compared to using random number generators, which can lead to longer processing times. However, the problem can be solved quickly using chaotic time series, making the computational overhead of chaotic maps negligible. Chaos systems with differential equations that need to be solved numerically are examples of such systems; however, this paper does not use these types of systems.

According to the numerous tests and experiments that were conducted with and without the presence of chaos, we did not encounter the phenomenon of overfitting caused by the certainty in chaos. In this paper, the deterministic property of chaos has not been used, and therefore there is no possibility of overfitting due to the certainty of chaotic sequences, here to generate the chaotic number every time by giving a random number to the chaotic system, its generated number is used. In this method, the process of selecting an action and the ε-greedy method occurs outside the network, and the type of number used for selecting an action (random number or chaotic number) does not affect the network training process, we only expect all actions to be tested multiple times based on the properties of chaos and the best action to be chosen.

Chaotic numbers produced in this paper can be considered as random numbers but with more random properties, which has been proven in tests and experiments. Due to the property of generated numbers, we have made more random choices and compared to the use of random numbers in the ε-greedy method, the algorithm converges earlier and the problem is learned sooner; Therefore, this method is more efficient than the traditional method. The generated chaotic numbers can be easily and without worry used instead of random numbers because each of the generated chaotic numbers is independent from the other numbers and the numbers are not dependent on each other.

8. Conclusion

For the first time in this paper, the use of chaos in deep reinforcement learning is presented in order to balance between exploration and exploitation. The proposed method was discussed and evaluated on the Lunar Lander environment and favorable results were obtained. Here, chaos was used both in the replay buffer to select samples for neural network training, and in the action selection part with the help of the ε-greedy method, which instead of random numbers, generated chaotic numbers were used. The results of using chaos in choosing the action well and much better than the traditional mode has been able to learn the problem and win more awards. Also, chaotic selection of samples in replay buffer was better than random selection of samples, and the combined use of chaos is more effective in both parts.

Given the variety and diversity of chaotic systems and somewhat different behaviors of some chaotic systems in different environments, selecting and deploying the desired system for a specific application can be challenging; although, considering the successful experience of chaos in numerous applications, generally, replacing random numbers with chaotic numbers is desirable in most cases.

Considering the lack of research in the field of using chaos in deep reinforcement learning, this topic could be a suitable area of work. It is possible to examine the effect of using other chaotic systems or their combination in this field. Additionally, exploring the development of the proposed method in other deep reinforcement learning architectures such as actor-critic structures or other reinforcement learning methods can be considered.

9. References

[۱] ولی درهمی، فریناز اعلمیان، محمدباقر دولتشاهی، «یادگیری تقویتی»، انتشارات دانشگاه یزد، ۱۳۹۶.

[۲] سید علی خوشرو، سید حسین خواسته، «افزایش سرعت فرآیند یادگیری DQN با مکانیزم آثار شایستگی»، مجله کنترل، جلد ۱۴، شماره ۴، صفحات ۲۳-۱۳، ۱۳۹۹.

[3] P. Ladosz, L. Weng, M. Kim, H. Oh, "Exploration in deep reinforcement learning: A survey", *Information Fusion*, vol. 85, pp. 1-22, 2022.

[4] H. Khodadadi, A. Zandvakili, "A New Method for Encryption of Color Images based on Combination of Chaotic Systems", *Journal of AI and Data Mining*, vol. 7, no. 3, pp. 377-383, 2019.

[5] R.B. Naik, U. Singh, "A review on applications of chaotic maps in pseudo-random number generators and encryption", *Annals of Data Science*, vol. 11, no. 1, pp. 25-50, 2024.

[6] H. Liu, A. Kadir, Y. Li, "Audio encryption scheme by confusion and diffusion based on multi-scroll chaotic system and one-time keys", *Optik*, vol. 127, no. 19, pp. 7431-7438, 2016.

[7] M.S Azzaz, M.A. Krimil, "A new chaos-based text encryption to secure gps data", In 2018 International Conference on Smart Communications in Network Technologies (SaCoNeT), October 2018, Algiers, Algeria , pp. 294-299.

[8] H. Xu, X. Tong, X. Meng, "An efficient chaos pseudo-random number generator applied to video encryption", *Optik*, vol. 127, no. 20, pp. 9305-9319, 2016.

[9] K. Chen, B. Xue, M. Zhang, F. Zhou, "Novel chaotic grouping particle swarm optimization with a dynamic regrouping strategy for solving numerical optimization tasks", *Knowledge-Based Systems*, 194, 105-123, 2020.

[۱۰] مجید محمدپور، حمید پروین، «الگوریتم ژنتیک آشوب گونه مبتنی بر حافظه و خوشه بندی برای حل مسائل بهینه سازی پویا»، مجله مهندسی برق دانشگاه تبریز، جلد ۴۶، شماره ۳، صفحات ۲۹۹-۳۱۸، ۱۳۹۵.

[۱۱] صمد نجاتیان، وحیده رضایی، حمید پروین، «ارائه یک الگوریتم چندجمعیتی مبتنی بر ازدحام ذرات برای حل مسائل بهینه‌سازی پویا»، مجله

مهندسی برق دانشگاه تبریز، جلد ۴۸، شماره ۳، صفحات ۱۴۰۵-۱۴۲۳، ۱۳۹۷.

[12] Z. Liang, Z. Xiao, J. Wang, L. Sun, B. Li, Y. Hu, Y. Wu, "An improved chaos similarity model for hydrological forecasting", *Journal of Hydrology*, vol. 577, pp. 123-133, 2019.

[13] Z. Hua, Y. Zhou, "Exponential chaotic model for generating robust chaos", *IEEE transactions on systems, man, and cybernetics: systems*, vol. 51, no. 6, pp. 3713-3724, 2019.

[14] J.T. Chien, P.C. Hsu, "Stochastic curiosity maximizing exploration", In 2020 International Joint Conference on Neural Networks (IJCNN), July 2020, Glasgow, UK, pp. 1-8.

[15] T. Lin, A. Jabri, "MIMEx: intrinsic rewards from masked input modeling", arXiv preprint arXiv:2305.08932, 2023.

[16] V. Derhami, V.J. Majd, M.N. Ahmadabadi, "Exploration and exploitation balance management in fuzzy reinforcement learning", *Fuzzy sets and systems*, vol. 161, no. 4, pp. 578-595, 2010.

[17] B.H. Abed-alguni, "Action-selection method for reinforcement learning based on cuckoo search algorithm", *Arabian Journal for Science and Engineering*, vol. 43, no. 12, pp. 6771-6785, 2018.

[18] A. Ecoffet, J. Huizinga, J. Lehman, K.O. Stanley, J. Clune, "First return, then explore", *Nature*, vol. 590, no. 7847, pp. 580-586, 2021.

[19] M. Usama, D.E. Chang, "Learning-driven exploration for reinforcement learning", In 2021 21st International Conference on Control, Automation and Systems (ICCAS), (2021, October).*)* (pp. 1146-1151). IEEE.

[20] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, and Y. Tassa, "Safe exploration in continuous action spaces", arXiv preprint arXiv:1801.08757, 2018.

[21] T.G. Karimpanal, S. Rana, S. Gupta, T. Tran, S. Venkatesh, "Learning transferable domain priors for safe exploration in reinforcement learning", In 2020 International Joint Conference on Neural Networks (IJCNN),July 2020, Glasgow, UK, pp. 1-10.

[22] A. Zhaikhan, A.H. Sayed, "Graph Exploration for Effective Multiagent Q-Learning", *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1-12, 2024.

[23] K. Morihiro, T. Isokawa, N. Matsui, H. Nishimura, "Effects of chaotic exploration on reinforcement learning in target capturing task", *International Journal of Knowledge-based and Intelligent Engineering Systems*, vol. 12, no. 5-6, pp.369-377, 2008.

[24] K. Morihiro, T. Isokawa, N. Matsui, H. Nishimura, "Reinforcement learning by chaotic exploration generator in target capturing task", In International Conference on Knowledge-Based and Intelligent Information and Engineering Systems, 2005, Melbourne, Australia, pp. 1248-1254.

[25] K. Morihiro, N. Matsui, H. Nishimura, "Effects of chaotic exploration on reinforcement maze learning", In International Conference on Knowledge-Based and Intelligent Information and Engineering Systems, (pp. 833-839). Springer, Berlin, Heidelberg.

[26] K. Morihiro, N. Matsui, H. Nishimura, "Chaotic exploration effects on reinforcement learning in shortcut maze task", *International Journal of Bifurcation and Chaos*, vol. 16, no. 10, pp. 3015-3022, 2006.

[27] A.B Potapov, and M.K Ali, "Learning, exploration and chaotic policies", *International Journal of Modern Physics C*, vol. 11, no. 07, pp. 1455-1464, 2000.

[28] H. Khodadadi, V. Derhami, "Improving Speed and Efficiency of Dynamic Programming Methods through Chaos", *Journal of AI and Data Mining,* vol. 9, no. 4, pp. 487-496, 2021.

[29] B. Zarei, M.R Meybodi, "Improving learning ability of learning automata using chaos theory", *The Journal of Supercomputing*, vol. 77, no. 1, pp. 652-678, 2021.

[30] X. Zhang, Y. Cao, "A novel chaotic map and an improved chaos-based image encryption scheme", *The Scientific World Journal*, Article ID 713541, 2014.

[31] H. Van Hasselt, A. Guez, D. Silver, "Deep reinforcement learning with double q-learning", In Proceedings of the AAAI conference on artificial intelligence, (Vol. 30, No. 1), Phoenix, Arizona USA.

[32] T. Schaul, J. Quan, I. Antonoglou, D. Silver, "Prioritized experience replay", arXiv preprint arXiv:1511.05952, 2015.

[33] Z. Wang,T. Schaul, M. Hessel, H. Hasselt, M .Lanctot, N. Freitas, "Dueling network architectures for deep reinforcement learning", In International conference on machine learning, June 2016, PMLR, pp. 1995-2003.

[34] RS. Sutton, AG, Barto, "Reinforcement learning: An introduction", 2nd Ed, The MIT Press, London, 2018.

[35] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, "Open ai gym", ArXiv:1606.01540, 2016.