International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(10s) 2026 pp. 133-146
https://doi.org/10.70917/ijcisim-2026-3573
Article

# Evaluating Deep Reinforcement Learning Techniques for Sequential Decision-Making in Arcade Games

**Arti Deshpande[1*], Bhushan Jadhav[2], Eesha Karnani[3], Vanshika Kaurani[4], Hiren Khatri[5]**

Thadomal Shahani Engineering College, Mumbai, India, email: arti.deshpande@thadomal.org

**Abstract:**Reinforcement Learning (RL) is a powerful approach for training agents to make decisions in complex and dynamic environments, particularly those with high-dimensional visual input. However, understanding of architectural changes affect learning performance, stability and efficiency is an important research challenge. This paper compares the five value-based reinforcement learning algorithms, namely, SARSA, Deep Q-Network (DQN), Double Deep Q-Network (DDQN), Dueling DQN, and Dueling DQN with Prioritized Experience Replay (PER) on the MsPacman-v5 environment of Arcade Learning Environment (ALE). To compare fairly, all deep learning models were trained using a consistent pixel-based preprocessing pipeline of frame skipping, grayscale, resizing, and frame stacking while SARSA was represented in a discretized form based on the same environment to ensure fair comparison.

The algorithms were evaluated using metrics such as maximum score, average score, convergence behavior, and training efficiency. The results show that architectural improvements play a significant role in enhancing learning performance. Dueling DQN achieved the best performance, reaching a maximum score of 4650 and an average score of 1225.7, demonstrating its ability to learn more effective state representations. DDQN also showed strong performance, achieving a maximum score of 4000 and improving average performance by 41.7% compared to standard DQN, confirming its effectiveness in reducing overestimation bias. In contrast, SARSA demonstrated stable but limited performance due to its simplified state representation. Overall, the findings highlight the importance of architectural design in improving reinforcement learning performance in complex decision-making environments.

**Keywords:** Reinforcement Learning, Adaptive AI system, Convolutional neural network, Game AI benchmarking.

## 1. INTRODUCTION

The field of Reinforcement Learning (RL) is a machine learning paradigm in which an agent learns optimal policies through interaction with a dynamic environment [1]. In contrast to supervised learning, which is limited by the presence of labeled data, RL agents engage in a trial-and-error exploration policy to optimize the long-term cumulative reward. This approach is particularly effective in sequential decision-making problems where the aftermath of an act cannot be directly realized [2]. Temporal difference learning and policy optimization constitute the theoretical background of reinforcement learning that has a well-established base in classical literature, including a seminal work by Sutton and Barto [3] [9].

The development of RL has been characterized by a major transition of tabular means to the use of deep learning-based approximations. Conventional algorithms, including SARSA (State-Action-Reward-State-Action), are based on the use of on-policy temporal difference updates to compute action-value functions [4]. These methods are usually not high-dimensional state space methods, as they tend to suffer the curse of dimensionality. Mnih et al. introduced the Deep Q-Network (DQN) which introduced a breakthrough by combining Q-learning with Convolutional Neural Networks (CNNs) to enable agents to learn on raw pixels and perform as well as humans on the tasks of Atari games [4], [5]. Recent studies have since extended deep RL models to include better feature representation and attention-based models, including the Weight Sharing Attention (WSA), to provide better state representation with computational efficiency [6].

Video games, particularly those in the Arcade Learning Environment (ALE), serve as effective benchmarking platforms of reinforcement learning algorithms because they present a wide array of challenges

and have a rich visual input that is highly dimensional [4], [5]. One of them is Ms. Pac-Man which is a variant of a challenge that has partially observable state information, stochastic adversarial agents and requires the agent to cover the short term survival as well as the long term maximization of rewards [7]. The recent research has focused on the significance of enhanced training stability and replay buffer controls, such as priority techniques focusing on fresher and more pertinent experiences to hasten the convergence of learning [8]. Also, the current study of reinforcement learning still looks into the issues of architectural scaling and optimization methods to improve the performance and generalization skills on complex environments [9].

A number of architectural enhancements have been suggested to solve the shortcomings in standard DQN. To overcome the overestimation bias in estimating Q-values, Double DQN (DDQN) was proposed, which leads to a more stable learning process and better performance [14]. Double Deep Q-Networks (DDQN) address the overestimation bias present in standard DQN by decoupling action selection and evaluation, leading to more accurate value estimation and improved learning performance in complex environments. In the same manner, the Dueling DQN architecture consists of the decoupling of the computation of both the state value and the action advantage, enabling the network to acquire more resilient representations of environment dynamics [6]. Although increasing network depth and architectural complexity has been shown to increase performance in specific tasks with reinforcement learning, recent comparative studies have shown that architectural advances and experience replay optimization can yield more reliable performance improvement in terms of learning stability and sample efficiency [10] [11].

Despite these advancements, there remains limited comparative analysis evaluating SARSA, DQN, DDQN, and Dueling DQN under identical hyperparameter settings in complex, maze-based environments such as Ms. Pac-Man. Most previous research has been on the optimization of algorithms, not on comparing and evaluating them under fixed experimental conditions [12]. To fill this gap, in this research the authors compare five reinforcement learning algorithms: SARSA, DQN, DDQN, Dueling DQN and Dueling DQN with Prioritized Experience Replay (PER) in the MsPacman-v5 environment, which is a classic game. These algorithms are evaluated using performance metrics such as maximum score, convergence behavior, stability, and training efficiency [13]. The objective is to provide a deeper understanding of how architectural innovations, including overestimation bias correction and value-advantage decomposition, influence agent performance in high dimensional sequential decision making environments.

## 2. Literature Review

Many of the recent deep reinforcement learning algorithms are based on value-based reinforcement learning techniques. The initial methods that included Q-learning and SARSA used tabular representations to approximate action-value functions through temporal difference learning. Watkins proposed Q-learning, which is an off-policy algorithm that can learn the best policies regardless of the exploration policy of the agent [24]. Conversely, SARSA is an on-policy algorithm and updates its value estimates depending on the actual actions of the agent, which causes it to behave more conservatively and more stable during the learning process [9]. These methods were formalized and became the theoretical framework of temporal difference learning, policy analysis, and control in sequence decision-making systems by Sutton and Barto [9].

Although these tabular approaches were shown to be effective in small scale and discrete state spaces, when applied in high-dimensional observation spaces they were found to be ineffective. The exponential growth of possible states made it impractical to store and update value functions using lookup tables, creating the need for function approximation techniques capable of generalizing across similar states.

The deep neural network combined with reinforcement learning was a significant breakthrough to surmount the tabular methods. Deep Q-Network (DQN) by Mnih et al. is an algorithm based on convolutional neural networks (CNNs) and employs these networks to estimate the Q-function using raw pixel input directly [15]. The method allowed agents to acquire significant spatial characteristics and to generalize in visually complicated situations. These deep reinforcement learning methods typically rely on standardized preprocessing pipelines, including grayscale conversion, frame resizing, and temporal frame stacking, to efficiently extract spatial and temporal features from high-dimensional visual input.

Subsequent developments revealed that DQN was capable of being able to perform like humans in a variety of Atari 2600 games when given only visual input and reward data [5]. Experience replay and target networks were some of the major advancements that enhanced the stability of training by minimizing correlations of one observation with another and stabilizing the temporal difference updates. These works made deep reinforcement learning a powerful framework to solve high-dimensional control tasks and made Atari environments, such as Ms. Pac-Man, a standard benchmark to assess algorithm performance [17].

Although successful, DQN was not stable because it had the overestimation bias in the action-value estimation. This problem is due to the fact that the same network is used in the process of choosing and evaluating

actions, leading to excessively optimistic value estimates. To overcome this weakness, Van Hasselt et al. presented Double Deep Q-Learning (DDQN) that does not combine the content to select actions and evaluate them, but instead uses different networks [14].

Double Deep Q-Learning (DDQN), introduced by Van Hasselt et al. [14], addresses overestimation bias in standard DQN by separating action selection and action evaluation using two neural networks. This decoupling leads to more accurate value estimation and better learning of policy. Prior work has shown that DDQN can achieve better performance and stability in several Atari environments, including those with a high visual complexity where the value estimation must be accurate.

New architecture enhancements were brought to increase the efficiency of learning and quality of representation. Wang et al. [6] proposed separating the state value function and action advantage function in the neural network in Dueling Deep Q-Network. This decomposition allows the model to learn from the states without needing to know the actions which results in better generalization and learning efficiency. The two-architectures also enable more stable and faster convergence in multi-action scenarios with similar outputs like a navigation task. This architecture is shown to be more efficient in learning leading to improved performance on many Atari environments compared to the baseline DQN and faster convergence.

Experience replay plays a critical role in stabilizing deep reinforcement learning since it enables agents to replay previous experiences. Random sampling uniformly, however, is not able to differentiate between informative and less useful transitions. Schaul et al. proposed Prioritized Experience Replay (PER) that gives greater sampling probabilities to larger temporal difference errors transitions [18] [19].

Such prioritization allows the agent to pay more attention to more informative and surprising experiences, increasing learning and convergence. The recent study by Obando-Ceron et al. added to this idea by showing that a more recent experience may greatly enhance learning effectiveness and flexibility under dynamic conditions [22]. The developments demonstrate the relevance of experience sampling strategies to enhance the performance in deep reinforcement learning [20] [21].

In recent years, there has been research on enhancing the sample efficiency, stability, and the generalization ability of deep reinforcement learning algorithms in complex environments [23]. The Rainbow architecture by Hessel et al. combines various enhancements, such as Double Q-learning, prioritized experience replay, and dueling networks, into a single architecture, with significant performance improvements in all benchmarks on the Atari [25]. This work demonstrated that combining complementary architectural enhancements can significantly improve learning robustness. In parallel, distributional reinforcement learning has emerged as an important advancement, where agents learn full return distributions rather than expected values. Bellemare et al. showed that this approach improves training stability, enhances value estimation accuracy, and provides richer learning signals in high-dimensional environments [26].

Recent research has been devoted to better representation learning and sample efficiency with auxiliary objectives and architectural improvements. Schwarzer et al. have introduced Data-Efficient Rainbow to consider self-supervised auxiliary tasks that enhance the representation of features and speeds up learning using limited interactions [27]. Equally, Yarats et al. have shown that the effective network structures, normalization strategies, and feature derivation mechanisms are able to promote the stability of the training and speed of convergence in reinforcement learning systems greatly [28]. The progresses highlight the need to rely on representation learning and architecture design in improving the performance of the reinforcement learning algorithm in sparse-rewarded environments and nontrivial visual environments such as those with Atari games.

Experience replay mechanisms are also subject to lots of development with the aim of making learning more efficient. Obando-Ceron and Castro showed that the recent and informative experiences are crucial for prioritizing convergence rates and overall performance in the Atari settings; where the agents were able to focus on more relevant transitions [22]. Studies in recent years have revealed that policies based on DQN, such as Double DQN and Dueling DQN, are still effective reinforcement learning methods. These algorithms can learn effectively with relatively stable training, and have good computational efficiency, which will be used to solve complex decision-making problems [29]. Advancements like better architectures, experience replay, and state-value representation have been key in driving the ongoing advancement of deep reinforcement learning. Even with all these developed algorithms, few studies have directly compared these algorithms under the same experimental conditions. In particular, there is relatively little research that has examined them with the identical preprocessing methods, hyperparameters and training setting for a visually complex, maze-type game like MsPacman-v5. There has been a focus in the majority of the existing work on optimising the performance of a single algorithm, without a common evaluation framework that allows for the comparison of multiple approaches. To overcome this drawback, the present study compares systematically the SARSA, DQN, DDQN, Dueling DQN and Dueling DQN with Prioritized Experience Replay algorithms under the same training setup. The same

experimental design allows them to be compared fairly with respect to their learning ability, convergence behaviour, stability and overall performance.

In order to tackle these problems, the proposed study compares these algorithms systematically in MsPacman-v5 environment, and discusses the performance efficiency and convergence property of these algorithms in comparison to each other. In this work, we explored the performance of the algorithms in the same experimental environment, and examined the differences in their architecture and performance in the same context, which will aid in understanding the performance of reinforcement learning in complex real-time environments.

## 3. Methodology

The proposed experiments have been performed on the MsPacman-v5 environment of the Arcade Learning Environment (ALE) as part of the OpenAI Gym framework, which offers a standard interface through which reinforcement learning research and results can be compared and contrasted [16]. The environment gives observations in the form of raw RGB frames of size 210 x 160 x 3, which are the entire visual state of the game. All the implemented algorithms relied on these observations as the main source of the state information which guaranteed that the course of interaction with the environment will be similar, and it will be possible to compare the performances in a meaningful way. Environment dynamics are episodic, with the agent operating on the environment, by choosing discrete actions to get the highest reward, and the particular episode ends when the agent loses all lives, or the level goals are achieved.

In deep reinforcement learning algorithms, such as DQN, DDQN, Dueling DQN or Dueling DQN with Prioritized Experience Replay, the raw RGB frames were fed straight to the convolutional neural networks, and the models would then learn to identify the spatial characteristics and the mechanics of the game. Nonetheless, as SARSA is tabular reinforcement learning, and no longer effectively scales to the image inputs of high dimensions, a discrete representation of the state was built based on the same RGB frames. The process of discretization identified the important game elements like agent position, ghost position and surrounding pellets by dividing the RGB frame into a structured format of a grid. This method retains the necessary environmental data and down samples state dimensions to a feasible level of size to learn using tabular methods. Significantly, the discretized state representation was based directly on the same RGB images given by the environment, which was consistent in all algorithms in terms of environmental data and had to fit within the respective computational and architectural requirements of the algorithms.

The MsPacman-v5 setting can be considered to be partially observable, stochastic and dynamic, and demands the agent to develop adaptive behavior in the environment, which has adversarial agents and delayed rewards. It has discrete action space consisting of 9 predefined moves, episodic structure and high-dimensional visual observations, which makes it a difficult and popular benchmark to test reinforcement learning algorithms. This setting offers an appropriate test environment in which the relative performance, stability, and learning efficiency of classical and deep reinforcement learning methods can be studied.

The hardware setup for training the reinforcement learning agent includes an Intel i5 processor (4.40 GHz), 16 GB of RAM, and an NVIDIA RTX 3060 GPU with 12 GB VRAM, ensuring smooth execution of deep learning models with efficient parallel processing The software stack includes PyTorch for deep learning, model training data; Gymnasium (an improved fork of OpenAI Gym) for reinforcement learning environments; TensorBoard for performance visualization; and Matplotlib for plotting and analyzing data. Additionally, NVIDIA CUDA Toolkit is utilized to enable GPU acceleration in PyTorch, significantly improving training speed and efficiency by offloading computations to the RTX 3060. This combination of powerful hardware and optimized software libraries ensures effective reinforcement learning model development and experimentation.

### *3.2 Preprocessing*

A standardised preprocessing pipeline was used to make sure that the computations were efficient and the features extracted would be useful in Deep Q-Networks. The preprocessing procedures of the Deep Learning agents (DQN variants) and the Tabular agent (SARSA) have minor variations to meet the input requirements of their agents.

#### 3.2.1 Pixel-Based Preprocessing

In all Deep Q-Network versions, the raw RGB images of the Atari emulator are then processed through the following transformation chain, according to the protocols defined by Mnih et al. (2015) [5]:

1. No-Op Reset: At the start of each episode, the agent executes a random number of "No-Operation" actions (sampled uniformly between 1 and 30).This adds stochasticity to the starting state, making the agent unable to memorize a predetermined starting sequence (trajectory overfitting).

2. Frame Skipping and Max-Pooling: The agent repeats a selected action for $k=4$ consecutive frames. In order to remove flickering artifacts due to the Atari hardware (which only displays a portion of sprites per frame) the maximum of the previous two frames in the skip window, element-wise, is calculated.
3. Grayscale Conversion and Resizing: The raw 210 x 160 x 3 RGB frame is converted to grayscale and down-sampled to an 84 x 84 matrix. This saves on the input dimensionality by about 97 percent, which hugely decreases the computational burden of the Convolutional Neural Network (CNN).
4. Frame Stacking: Frame stacking is used to record temporal dynamics (e.g. direction and speed of ghosts) by stacking the four most recent processed frames along the channel dimension. St is the final input state that is a (4, 84, 84) -shaped tensor.

### *3.3 Proposed System Architecture*

Figure 1. outlines a unified reinforcement learning pipeline supporting Dueling DQN, DQN, DDQN, and SARSA algorithms. Initialization varies by method: SARSA employs a tabular Q-value representation, while the others utilize deep Q-networks (with Dueling DQN incorporating separate value and advantage streams) along with a replay buffer. Action selection is governed by an ε-greedy policy. SARSA performs on-policy temporal difference updates, whereas the others adopt off-policy learning by storing transitions in the replay buffer and sampling mini-batches. DDQN mitigates Q-value overestimation by decoupling action selection (via the online network) from evaluation (via the target network), while DQN and Dueling DQN use target networks for both. The main Q-network is updated through gradient descent, and the target network is periodically synchronized. Epsilon is decayed post-episode to gradually shift from exploration to exploitation.

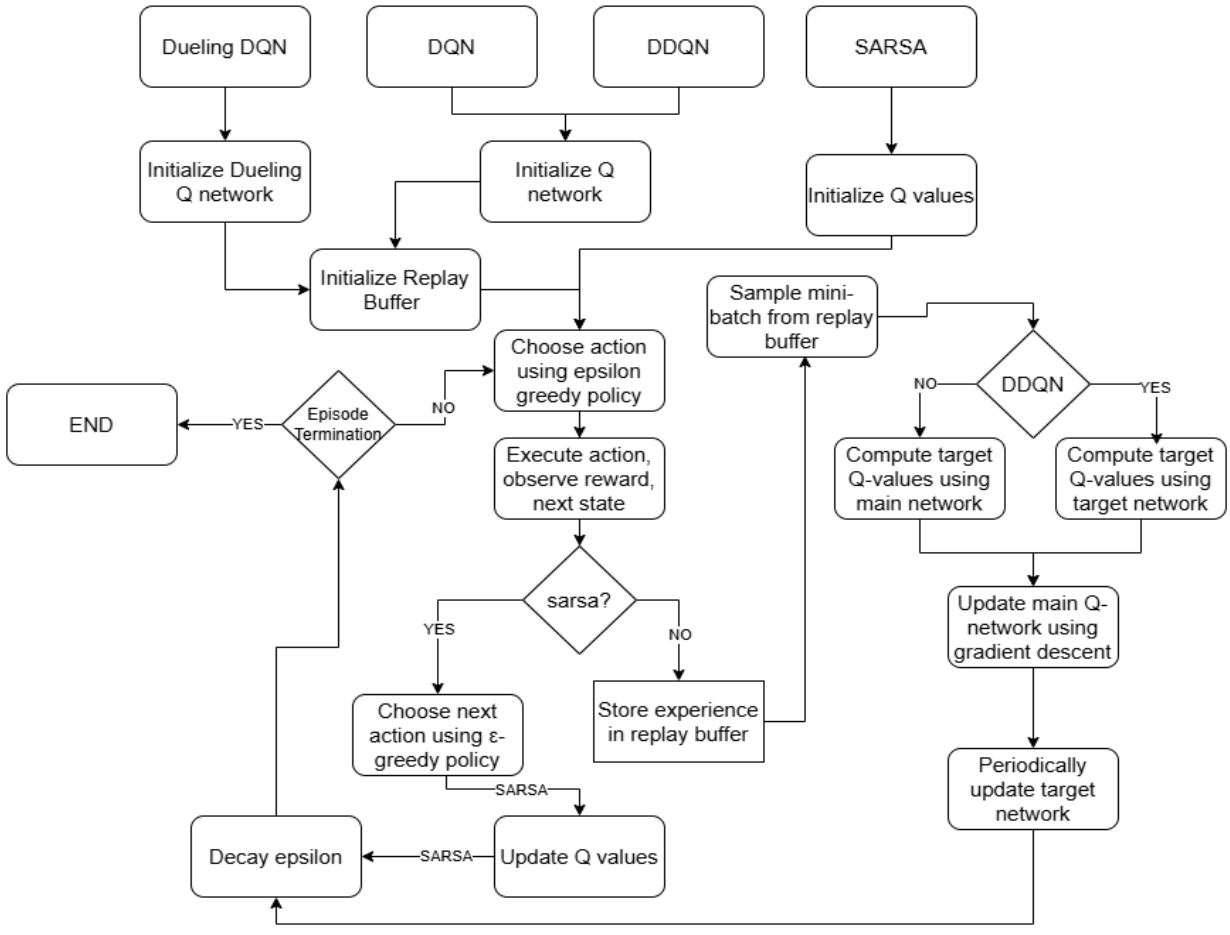


**Figure 1:** Flow diagram of the proposed system

### *3.4 Algorithms*

This paper trains and tests five value-based reinforcement learning algorithms, including SARSA, Deep Q-Network (DQN), Double Deep Q-Network (DDQN), Dueling DQN, and Dueling DQN with Prioritized Experience Replay (PER) to examine their performance in the MsPacman-v5 game. These algorithms reflect the important developments in the history of reinforcement learning, starting with the classical tabular temporal-difference algorithms, all the way to more modern techniques based on deep neural networks that include a combination of architectural and experience replay improvements. The following subsections describe the theoretical formulation, preprocessing pipeline, and implementation details of each algorithm.

### 3.4.1 SARSA (State-Action-Reward-State-Action)

In this study, the SARSA (State-Action-Reward-State-Action) algorithm is implemented to train an agent for playing *Ms. Pac-Man* while utilizing a discretized state representation. SARSA is an on-policy reinforcement learning algorithm that updates the Q-value for a state-action pair using the formula given in equation (1)

$$Q(S,A) = Q(S,A) + \alpha(R + \gamma Q(S',A') - Q(S,A)) \quad (1)$$

Where Q(S,A) is the action-value function, α is the learning rate, R is the reward obtained, γ is the discount factor, and (S′,A′) represents the next state-action pair selected using the current policy. Unlike off-policy algorithms like Q-learning, SARSA follows the same policy for both learning and action selection, making it more stable in environments where exploration is necessary [9].

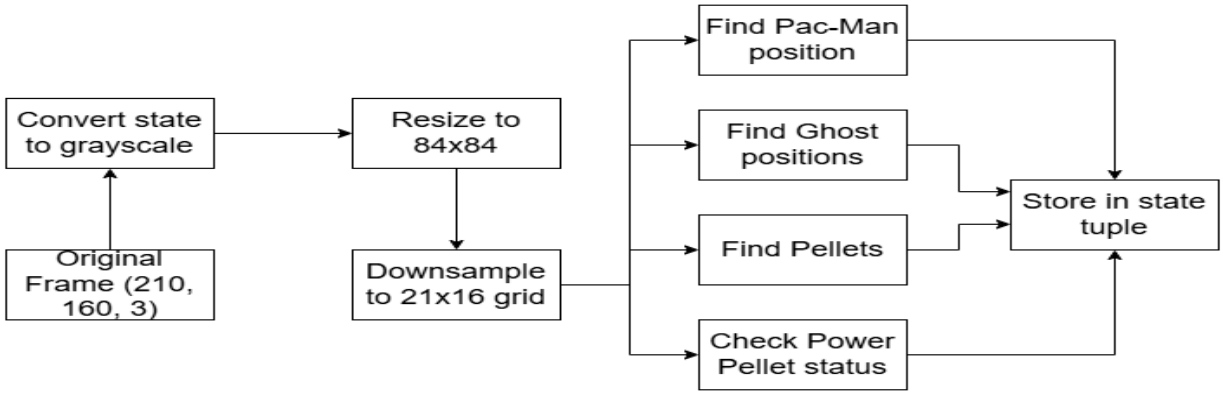


**Figure 2:** Steps in preprocessing for SARSA

Figure 2 illustrates the steps for preprocessing the game screen in the SARSA implementation. initially discretize the 210 × 160 pixel frame into a 21 × 16 grid, where each cell corresponds to a 10 × 10 pixel region. It reduces computational complexity while preserving critical game information. The discretized state includes Pac-Man's position (grid coordinates), ghost positions, and nearby pellet locations within a predefined radius. Power pellet status is represented as a binary value indicating whether they are active. To preprocess the game screen, initially frames are converted to grayscale, resized to 84 × 84, and further down sampled to 21 × 16 using bilinear interpolation [8, 11]. Object detection methods based on color thresholds and contour analysis are used to extract relevant game entities. This structured representation enables SARSA to learn an optimal policy efficiently by focusing on key game dynamics rather than raw pixel values.

### 3.4.2 DQN

DQN was the first method to take raw pixels as input and effectively learn control strategies directly from raw data from sensors using reinforcement learning [4,15]. It integrates a neural network with classical Q-learning algorithm to approximate Q-values, the maximum expected (Q*) reward is defined in equation (2)

$$Q*(s,a) = E[r + \gamma max_{a'} Q*(s_{t+1}, a') \mid s_t = s, a_t = a] \quad (2)$$

To improve training stability, DQN minimizes the loss function as given in equation (3)

$$L(\theta_i) = E_{s,a,r,s'}[(y_i - Q(s,a;\theta_i))2] \quad (3)$$

where $yi = r + \gamma max_{a'} Q(s', a'; \theta_i)$ is the target value calculated using a separate target network to prevent divergence.

The policy network approximates the Q-function by taking input as state s and giving output as Q-values, enabling action to be selected with help of an ε-greedy strategy .The target network computes target Q-values. It is a delayed replica of the policy network, and is synchronized after regular intervals to stabilize training and prevent divergence.

This combination of CNN-based feature extraction, experience replay, and stable Q-learning updates enables the agent to achieve human-level performance in MsPacman.

### 3.4.3 DDQN

Double Deep Q-Network (DDQN) is an advanced version of the traditional DQN designed to address the problem of Q-value overestimation in reinforcement learning. In standard DQN, the target Q-value is computed using the formula given in equation (4)

$$Q_{target}(s, a) = r + \gamma max Q_{target}(s', a') \quad (4)$$

where the target network is responsible for choosing actions and estimating their value, often leading to overestimated Q-values. DDQN addresses this by separating the processes of action selection and evaluation using two networks: policy network and target network.

The DDQN update modifies the Q-value computation as given in equation (5)

$$Q_{target}(s, a) = r + \gamma Q_{target}(s', arg\ max Q_{policy}(s', a')) \quad (5)$$

The action to be performed by the agent in the environment is selected by the policy network. And the corresponding Q-Value for the action performed is evaluated by the target network. As two separate neural networks are used the risk of overoptimistic value estimates is minimized. As a result, learning is more reliable and consistent [14].

### 3.4.4 Dueling DQN

Dueling DQN uses a separate function to calculate the expected reward for being in a particular state s. And another function to calculate the difference between the value provided by available actions [6]. Since, in this method two separate streams are utilized, it permits the model to better evaluate states independently of specific actions, leading to improved policy learning in environments where some actions have little impact on the overall outcome. The key benefit of this approach is that it enables better generalization across actions, making learning more efficient, especially in scenarios with redundant or similar-valued actions.

Compared to DDQN, which addresses the overestimation bias of Q-values, Dueling DQN focuses on improving the representation of state values. While DDQN refines the Q-value estimation process, Dueling DQN enhances policy evaluation by learning which states are valuable without necessarily knowing the optimal action at each step. This allows it to identify the importance of different game states more effectively, which is particularly useful in high-dimensional environments like Atari games.

The Q-value function in Dueling DQN is computed as given in equation (6) $Q(s, a;\ \theta, \alpha, \beta) = V\ (s;\ \theta, \beta) + (A(s, a;\ \theta, \alpha) - \frac{1}{|A|}\sum_{a'} A\ (s, a'\ ;\ \theta, \alpha))$ (6)

Where

V(s;θ,β) represents the value function.

A(s,a;θ,α) is the advantage function.

Dueling DQN retains the three convolutional layers from traditional DQNs but introduces two fully connected streams:

1. Value Stream: Outputs a single scalar V(s).
2. Advantage Stream: Outputs |A| values, where |A| is the number of possible actions.

These two streams are then combined using the above formula to compute Q-values for each action.

### 3.4.5 Duel with prioritized experience buffer

Standard experience replay randomly samples past experiences, which may lead to inefficient learning as all transitions are treated equally. Prioritized Experience Replay (PER) improves upon this by assigning higher

priority to more significant experiences, measured using temporal difference (TD) error [19]. The probability of selecting an experience is given in equation (7)

$$P(i) = \frac{P^{\alpha}{}_i}{\Sigma_k P^{\alpha}{}_k} \quad (7)$$

where $Pi = |TDi| + \epsilon$ confirms that no transition is excluded from sampling. The parameter α controls the importance of prioritization (typically set between 0.6 and 1.0), and a correction term β compensates for bias in sampling, gradually increasing from 0.4 to 1.0 during training.

The integration of Prioritized Replay and Dueling DQN (PDD-DQN) resulted in significant performance gains compared to standard DQN. The agent not only learned to navigate the environment more effectively, but also avoided getting stuck in suboptimal states, a common issue with conventional DQNs.

### *3.5 Hyperparameter Configuration*

To compare the performance of all deep reinforcement learning algorithms, the equivalent set of hyperparameters was used during training wherever applicable. The values were chosen from the previous research studies, and were tuned in via experiments to guarantee stable learning and reliable convergence. SARSA is a tabular on-policy reinforcement learning algorithm that does not need experience replay, mini-batch training or target networks. The hyperparameters for each algorithm are shown in Table I.

**Table I:** Hyperparameter configuration

| Hyperparameter | SARSA | DQN | DDQN | Dueling DQN | Dueling DQN with Prioritized |
|---|---|---|---|---|---|
| **Learning Rate** | 0.001 | 0.00025 | 0.00025 | 0.00025 | 0.00025 |
| **Batch Size** | N/A (on-policy) | 32 | 32 | 32 | 32 |
| **Replay Buffer Size** | N/A (on-policy) | 100,000 | 100,000 | 100,000 | 100,000 |
| **Epsilon Decay** | Linear: 1.0 → 0.05 over 1M frames | Linear: 1.0 → 0.05 over 1M frames | Linear: 1.0 → 0.05 over 1M frames | Linear: 1.0 → 0.05 over 1M frames | Linear: 1.0 → 0.05 over 1M frames |
| **Optimizer** | N/A | Adam | Adam | Adam | Adam |
| **Target Network Update Frequency** | N/A | Every 1,000 steps | Every 1,000 steps | Every 1,000 steps | Every 1,000 steps |

### *3.6 Performance Metrics*

Three critical performance measures, namely maximum score, learning stability, and training time, were used to measure the effectiveness of the tested reinforcement learning algorithms. The max score is the score that the agent collects the most in an episode. This measure shows the agent's learning ability to make the correct

decisions and improve the game's performance. Algorithms with higher scores were deemed more effective in learning the game strategy. For learning stability, the reward trend over the training episodes was analyzed. The steady rise and fall of rewards showed a steady learning process, while sharp fluctuations in rewards indicated unstable learning. Convergence is crucial during stable convergence to confirm reliability of the learning process and the ability of the algorithm to yield consistent results. To test the computational efficiency of each algorithm, both the algorithms were trained. It is the aggregate of the time needed on the training phase before the end of the learning phase. The algorithms that took less time to train and had good performance were more computationally efficient. These three evaluation measures can be used together to have a complete picture of the algorithm performance. The learned policy is evaluated on the maximum score, the learning stability is evaluated based on the stability of the learning process, and the training time assesses the computational efficiency of the training process. These metrics allow to compare the SARSA and deep reinforcement learning algorithms fairly and systematically within the MsPacman-v5 Atari environment.

## 4. Results and Analysis

This study presents a comparative analysis of five reinforcement learning algorithms: SARSA, DQN, DDQN, Dueling DQN, and Dueling DQN with Prioritized Experience Replay (PER) using the MsPacman-v5 Atari environment. To ensure a fair evaluation, all algorithms were trained and evaluated under the same experimental conditions, including identical reward settings, an equal number of training episodes, and consistent environment configurations. Maintaining a common experimental setup allowed the learning performance, convergence behavior, training stability, and overall effectiveness of each algorithm to be compared accurately and objectively. Performance evaluation was conducted using the metrics defined including maximum score, average score, training time, stability, and sample efficiency.

### *4.1 Overall Performance Analysis*

Table II summarizes the peak and average scores achieved by each algorithm, along with the corresponding training time. Dueling DQN showed better performance in all of the metrics reaching the highest score of 4,650 and an average score of 1,225.7. This makes it possible that the hypothesis that separating state-value V(s) and the estimation of advantages A(s,a) enables the network to learn which states are worthwhile independent of the action that is taken and speeds up the optimization of the policy in the maze-like environment of Ms. Pac-Man.

**Table II:** Performance Summary for Ms Pacman Environment across 5 techniques

| Algorithms | Maximum score | Avg score | Training time (in hours) |
|---|---|---|---|
| SARSA | 1,820 | 702.7 | 0.73 |
| DQN | 3,350 | 797.0 | 2.94 |
| DDQN | 4,000 | 1,130.0 | 9.83 |
| DUEL DQN | 4,650 | 1,225.7 | 2.54 |
| DUEL DQN WITH PRIORITIZED BUFFER | 3,370 | 1,223.0 | 0.95 |

DDQN was ranked third and had a mean score of 1,130.0, which is an improvement of 41.7 percent of vanilla DQN. This finding confirms that Double Q-learning is not affected by overestimation bias, which is a typical problem in normal DQN whereby the values of action are always overestimated, thereby choosing poor policies.

Although SARSA produced the lowest mean score (702.7), the process was incredibly efficient with regard to calculations. This makes SARSA a possible resource-limited embedded system architecture, where the speed of training is of higher importance than the maximum achievable score.

The learning curves are shown in Figure 3. DQN variants exhibit a gradual increasing direction, but SARSA increases quickly within the initial few minutes but levels off early because of the constraints of linear functional approximation of complex spatial strategies.

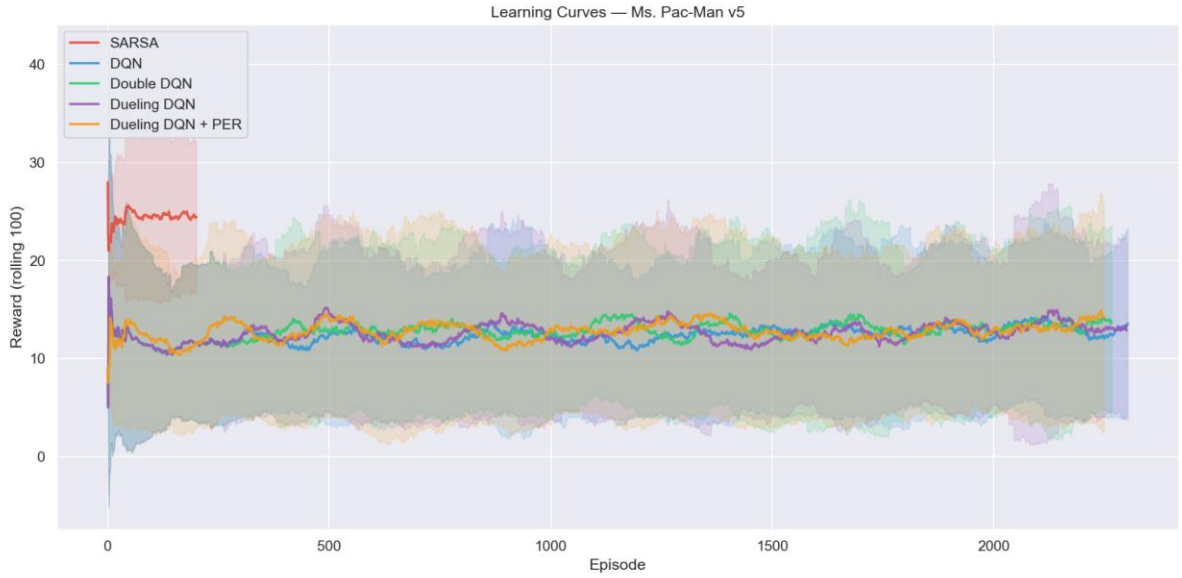


**Figure 3:** Learning curves of all five algorithms on Ms. Pac-Man v5. The x-axis represents training episodes, and the y-axis shows the rolling average episode reward (window = 100). Shaded regions denote ±1 standard deviation.

## *4.2 Learning Efficiency and Milestones*

To assess how quickly agents learned useful behaviors, we tracked the episode count required to reach specific score milestones.

To provide insight into sample efficiency, the number of episodes (Ep.) needed for each technique to reach the following score thresholds: 100, 200, 500, and 1200 are recorded as given in Table III. The row labeled as SARSA in Table 3 indicates the number of episodes required for the SARSA algorithm to achieve specific scores in the Pac-Man environment. It shows SARSA quickly reached scores up to 500 in the 150th episode, but did not achieve scores of 2000 or 4000.

**Table III:** Number of episodes required for an algorithm to reach a threshold score

| **Score** / **Technique** | **100** | **200** | **300** | **500** | **1000** | **2000** | **4000** |
|---|---|---|---|---|---|---|---|
| SARSA | Ep. 50 | Ep. 50 | Ep. 100 | Ep. 150 | Ep. 150 | | |
| DQN | Ep. 50 | Ep. 50 | Ep. 450 | Ep. 450 | Ep. 450 | Ep. 600 | |
| DDQN | Ep. 50 | Ep. 50 | Ep. 450 | Ep. 500 | Ep. 500 | Ep. 750 | Ep. 1050 |
| Dueling DQN | Ep. 50 | Ep. 50 | Ep. 600 | Ep. 600 | Ep. 600 | Ep. 950 | Ep. 1150 |
| Dueling DQN with prioritised buffer | Ep. 50 | Ep. 50 | Ep. 400 | Ep. 400 | Ep. 450 | Ep. 550 | |

Dueling DQN + PER was the most effective in the mid-range, with 2,000 points at the 550th episode. This confirms the theory according to which Prioritized Experience Replay can help to learn faster because the high-error transitions (surprising events) are more frequently replicated by the agent. Although PER was quicker in the short term, only Dueling DQN and Double DQN had the stability to achieve the 4,000-point mark. This indicates that standard uniform sampling can be more effective at policy fine-tuning in the end phases of training despite the fact that prioritization accelerates convergence.

### *4.3 Stability Analysis*

Stability was measured using the standard deviation of evaluation scores and policy oscillation.

**Table IV:** Stability Metrics

| Algorithm | Standard Deviation | Oscillation |
|---|---|---|
| SARSA | 282.3 | 28.0 |
| DQN | 357.1 | 63.0 |
| DDQN | 439.2 | 66.0 |
| Dueling DQN | 446.0 | 79.0 |
| Dueling DQN with prioritised buffer | 388.9 | 66.0 |

Figure 4 provides a visual representation of the evaluation score distributions for each algorithm, offering a complementary view of the stability metrics in Table 4.

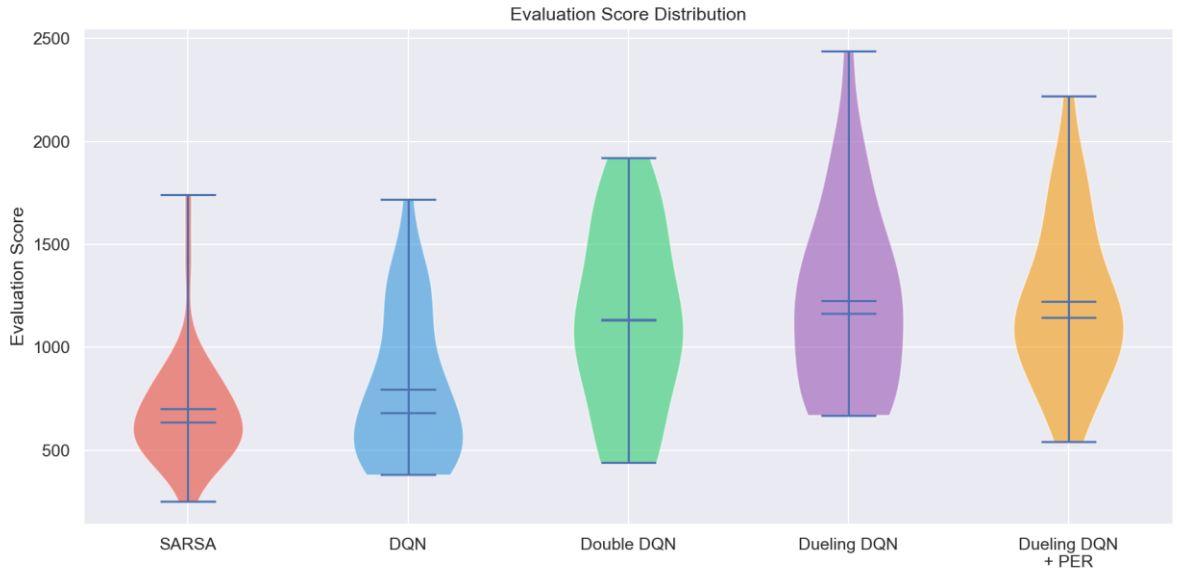


**Figure 4:** Evaluation score distributions across five evaluated algorithms

The width of each violin indicates the density of scores at that level. SARSA exhibits the tightest distribution (most stable), while Dueling DQN shows the widest spread (highest variance but also highest peak scores).

The stability of SARSA was the highest of all three metrics, and standard deviation (282.3) and oscillation score (28.0) were the lowest. This aligns with the desired characteristics of on-policy procedures, which are more likely to create more stable but perhaps less optimal policies since the update procedure is conservative.

Among the DQN variants, there is a clear trade-off between performance and stability. Dueling DQN, with the highest evaluation score (1,225.7) also had the highest instability with the most standard deviation (446.0) and oscillation (79.0). This indicates that the advantage stream of the Dueling architecture, although allowing more peak performance, introduces a larger difference between episodes in policy quality.

Prioritised Experience Replay on top of the Dueling architecture decreased the standard deviation of evaluation by 12.8 per cent (446.0 to 388.9), which means that prioritised sampling leads to a more active policy.

DQN showed moderate stability (Eval Std: 357.1, Oscillation: 63.0), occupying a middle ground between the conservative SARSA and the more volatile Dueling architectures. Although the theory of Double DQN is an improvement over the vanilla DQN in terms of overestimation reduction, it failed to show better stability than Vanilla DQN (Eval Std: 439.2 vs. 357.1), which indicates that the variance reduction of dual estimation focused on action-value accuracy, and not on policy consistency.

### *4.4 Statistical Significance analysis*

To go beyond descriptive comparisons, a set of statistical tests was performed on the 30 deterministic evaluation episodes realised at each of the terminal training checkpoints of every algorithm. Each pairwise test uses the Bonferroni adjustment of 10 concurrent comparisons to adjust the family-wise error rate.

Table 5 shows the average score of each algorithm and its 95 percent confidence interval, which has been calculated using the Student t-distribution.

**Table 5:** Mean Evaluation Scores with 95% Confidence Intervals

| Algorithm | N | Mean Score | Std Dev | 95% CI | Median |
|---|---|---|---|---|---|
| SARSA | 30 | 702.7 | 287.1 | [595, 810] | 635 |
| DQN | 30 | 797.0 | 363.2 | [661, 933] | 680 |
| DDQN | 30 | 1,130.0 | 446.7 | [963, 1,297] | 1,135 |
| Dueling DQN | 30 | 1,225.7 | 453.7 | [1,056,1,395] | 1,165 |
| Dueling DQN with prioritised buffer | 30 | 1,223.0 | 395.5 | [1,075,1,371] | 1,145 |

The findings of the t-tests (unequal-variance, two tailed) of all ten algorithm pairs, including the raw and Bonferonni-adjusted p-values, are given in Table 6.

**Table 6:** Pairwise Welch's t-Tests (Bonferroni-Corrected)

| Pair | t-stat | p (raw) | p (adj) | Sig. |
|---|---|---|---|---|
| SARSA vs DQN | −1.12 | 0.2693 | 1.0000 | ns |
| SARSA vs Double DQN | −4.41 | 0.0001 | 0.0006 | *** |
| SARSA vs Dueling DQN | −5.34 | <0.0001 | <0.0001 | *** |
| SARSA vs Dueling DQN with prioritised buffer | −5.83 | <0.0001 | <0.0001 | *** |
| DQN vs Double DQN | −3.17 | 0.0025 | 0.0249 | * |

| | | | | |
|---|---|---|---|---|
| DQN vs Dueling DQN | −4.04 | 0.0002 | 0.0017 | ** |
| DQN vs Dueling DQN with prioritised buffer | −4.35 | 0.0001 | 0.0006 | *** |
| Double DQN vs Dueling DQN | −0.82 | 0.4139 | 1.0000 | ns |
| Double DQN vs Dueling DQN with prioritised buffer | −0.85 | 0.3968 | 1.0000 | ns |
| Dueling DQN vs Dueling DQN with prioritised buffer | 0.02 | 0.9807 | 1.0000 | ns |

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$, ns = not significant.

The findings of t-tests performed in pairs to ascertain statistically significant differences in the mean evaluation scores within the five reinforcement learning algorithms are illustrated in table 6. Bonferroni correction was used to control the family-wise error rate to take into consideration a series of comparisons.

The analysis of SARSA and DQN did not present any statistically significant difference ($t = 1.12$, $p = 1.0000$), which means that the improvement of DQN in comparison with SARSA does not have a statistically reliable result. By contrast, SARSA achieved a much lower performance, as compared to all other advanced DQN-based architectures. The comparisons between SARSA and Double DQN, Dueling DQN and Dueling DQN with Prioritized Experience Replay was also significantly high ($p < 0.001$), which indicated the significant advantage of extreme deep architectural advancement compared to classical on-policy learning.

Under the deep reinforcement learning techniques, Double DQN was much better than normal DQN ($t = 3.17$, $p = 0.0249$), which proved the viability of reducing Q-value overestimation. Also, Dueling DQN as well as Dueling DQN with prioritised buffer had statistically significant increase in performance compared to vanilla DQN with stronger significance levels ($p < 0.01$ and $p < 0.001$ respectively). Those results suggest that value-advantage decomposition and prioritized sampling can be used to enhance the effectiveness of policy evaluation and learning.

In general, the statistical analysis has created a definite hierarchy of performance where the classical SARSA and usual DQN are always outperformed by the new advanced deep reinforcement learning frameworks.

The Jonckheere-Terpstra test of ordered alternatives was used with the a priori ranking of SARSA ≤ DQN ≤ Double DQN ≤ Dueling DQN ≤ Dueling DQN with prioritised buffer to formally test the hypothesis that performance increases monotonically with architectural sophistication.

The results of the test showed a Z-score of 5.78 ($p < 0.001$, one-tailed), which is statistically significant evidence that the architectural progression generates a monotonically increasing distribution in the evaluation performance. This confirms the main argument of this work: consecutive innovations in the value-based deep reinforcement learning architectures provide better and better performance in the domain of Ms. Pac-Man.

## 5. Discussion and conclusion

This study provides a comparative evaluation of five reinforcement learning algorithms: SARSA, DQN, DDQN, Dueling DQN, and Dueling DQN with Prioritized Experience Replay, using the challenging Ms. Pac-Man Atari environment as the evaluation platform. The objective is to examine and compare the learning performance, decision-making capability, and overall effectiveness of these algorithms in a complex game setting. The evaluation was conducted using standardized hyperparameters and training settings to ensure fairness across metrics such as maximum score, training stability, and computational efficiency. Dueling DQN became the most effective of all the methods, with the mean evaluation score of 1,225.7 and the maximum score of 4,650. This better ability, however, showed a negative correlation with stability because the Dueling agent was the most variable and the most oscillating of the policies. A monotonic trend in performance improvement of architectural sophistication was confirmed by formal statistical testing ($p < 0.001$.). The further development of standard DQN into Double DQN resulted in a massive statistically significant effect size ($d = 0.82$, $p = 0.0249$), which confirmed the applied significance of reducing the overestimation bias. The development of standard DQN to Double DQN and later on to Dueling DQN clearly illustrates that structural innovations, namely the reduction of overestimation

bias and splitting of state-value functions and advantage functions, are more profitable than sampling strategies itself. Although prioritized experience replay improved early learning speed by focusing on high-error transitions, it did not achieve the highest final performance compared to standard Dueling DQN. This is because prioritization can introduce sampling bias, reducing sufficient exploration of the full state space during later training stages. Overall, the findings demonstrate that architectural enhancements, such as value–advantage decomposition and prioritized experience replay, contribute significantly to improving learning stability, accelerating policy optimization, and achieving superior performance in complex state-space environments.

**References**


1. Tesauro G. Temporal difference learning and TD-Gammon. Commun ACM. 1995;38(3):58–68.
2. Naddaf Y, et al. Game-independent AI agents for playing Atari 2600 console games. University of Alberta; 2010.
3. Hershey D, Moody R, Wulfe B. Learning to play Atari games. Stanford University; 2015.
4. Mnih V, et al. Playing Atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602. 2013.
5. Mnih V, et al. Human-level control through deep reinforcement learning. Nature. 2015;518(7540):529–33.
6. Wang Z, et al. Dueling network architectures for deep reinforcement learning. arXiv preprint arXiv:1511.06581. 2015.
7. Rummery G, Niranjan M. On-learning using connectionist systems. Tech Rep CUED/F-INFENG/TR 166. Cambridge University Engineering Department; 1994.
8. Manoj T, Rohit J, Sai Sashank A, Praveena BJ. Deep reinforcement learning on Atari 2600. Int J Innov Sci Res Technol. 2021;6(4):487–9.
9. Sutton RS, Barto AG. Reinforcement learning: An introduction. MIT Press; 1998.
10. Zhao D, Wang H, Shao K, Zhu Y. Deep reinforcement learning with experience replay based on SARSA. In: IEEE Symposium Series on Computational Intelligence (SSCI). Athens; 2016. p. 1–6.
11. Nuzzo F. Unsupervised state representation pretraining in reinforcement learning applied to Atari games [Master's thesis]. KTH Royal Institute of Technology, Stockholm; 2020.
12. Bellemare MG, Naddaf Y, Veness J, Bowling M. The Arcade Learning Environment: An evaluation platform for general agents. J Artif Intell Res. 2013;47:253–79.
13. Gnanasekaran A, Feliu-Faba J, An J. Reinforcement learning in Pacman. Stanford University project report; 2017.
14. Van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double Q-learning. In: Proc AAAI Conf Artif Intell. arXiv:1509.06461. 2016.
15. Arulkumaran K, Deisenroth MP, Brundage M, Bharath AA. A brief survey of deep reinforcement learning. arXiv preprint arXiv:1708.05866. 2017.
16. Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J, Zaremba W. OpenAI Gym. arXiv preprint arXiv:1606.01540. 2016.
17. Sutton RS, McAllester D, Singh S, Mansour Y. Policy gradient methods for reinforcement learning with function approximation. In: Adv Neural Inf Process Syst. 2000. p. 1057–63.
18. Maddison CJ, Huang A, Sutskever I, Silver D. Move evaluation in Go using deep convolutional neural networks. In: Int Conf Learn Represent (ICLR); 2015.
19. Schaul T, Quan J, Antonoglou I, Silver D. Prioritized experience replay. In: Int Conf Learn Represent (ICLR); 2016.
20. Silver D, Huang A, Maddison CJ, Guez A, Sifre L, van den Driessche G, et al. Mastering the game of Go with deep neural networks and tree search. Nature. 2016;529(7587):484–9.
21. J. Obando-Ceron, A. Raffin, M. S. Castro, and A. Courville, "Combining Pre-Trained Models for Enhanced Feature Representation in Reinforcement Learning," arXiv preprint arXiv:2507.07197, Jul. 2025
22. J. Obando-Ceron and P. Castro, "Fresher Experience Replay: Prioritizing Recent Experience Improves Reinforcement Learning Performance," Applied Sciences, vol. 13, no. 1, 2023, doi: 10.3390/app13010123.
23. X. Nauman, M. Farebrother, and M. Bowling, "1000 Layer Networks for Self-Supervised Reinforcement Learning: Scaling Depth Enables Improved Goal-Reaching Capabilities," in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2024.
24. C. J. C. H. Watkins and P. Dayan, "Q-learning," Machine Learning, vol. 8, pp. 279–292, 1992.
25. M. Hessel et al., "Rainbow: Combining improvements in deep reinforcement learning," in Proc. AAAI Conf. Artificial Intelligence, vol. 32, no. 1, 2018.
26. M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in Proc. Int. Conf. Machine Learning (ICML), 2017.
27. M. Schwarzer et al., "Data-efficient reinforcement learning with self-predictive representations," in Proc. Int. Conf. Learning Representations (ICLR), 2021.
28. D. Yarats, I. Kostrikov, and R. Fergus, "Improving sample efficiency in model-free reinforcement learning," in Proc. Int. Conf. Machine Learning (ICML), 2021.
29. A. Terven and J. Cordova-Esparza, "A comprehensive review of deep reinforcement learning: Algorithms, applications, and future directions," Applied Sciences, vol. 15, no. 4, 2025.