AI AS A GAMER: APPLYING Q-LEARNING TO SPACE INVADERS

Kiran Palavalasa
University of Maryland
kpalaval@umd.edu

Onkar Prasanna Kher
University of Maryland
okher@umd.edu

***Abstract*—This project explores the application of Q-learning, a model-free reinforcement learning algorithm, to autonomously play and master the arcade game Space Invaders. Unlike traditional machine learning methods that require labeled data and explicit corrective feedback, Q-learning operates through a system of rewards and penalties, enabling an agent to learn optimal actions through trial and error. Our implementation centers on the development of a Q-table, where each entry represents an action-state pair with a value indicating the potential reward of taking that action in that state.**

The project's objectives were to configure and tune the Q-learning parameters such as the learning rate, discount rate, and exploration rate to enhance the learning efficacy and performance of the AI agent. The agent was trained over 5000 episodes, with each session providing insight into the strategy development and adaptation capabilities of the model.

Our results indicate significant learning and performance gains as the agent's experience accumulates, showcasing the effectiveness of Q-learning in environments where decision-making is critical. This study not only validates the utility of reinforcement learning in video game contexts but also suggests broader applications in scenarios requiring autonomous decision-making and strategy development.

I. Introduction

A. *Background and Context*

Reinforcement learning (RL) has emerged as a powerful paradigm in the field of machine learning, enabling agents to learn optimal behaviors through interactions with their environment. Unlike supervised learning, which relies on labeled data, Reinforcement Learning leverages the concept of rewards and penalties to guide the learning process. Among the various Reinforcement Learning algorithms, Q-learning stands out for its simplicity and effectiveness in solving complex decision-making problems without requiring a model of the environment.

The advent of Open-AI Gym has provided researchers and practitioners with a versatile toolkit to develop and benchmark Reinforcement Learning algorithms in a variety of environments, ranging from simple tasks to complex games like those in the Atari suite. These environments serve as standardized platforms to evaluate the performance of Reinforcement Learning agents, facilitating progress and innovation in the field.

B. *Problem Statement*

Space Invaders is a classic arcade game from the Atari 2600 series that presents a significant challenge for reinforcement learning due to its dynamic and partially observable nature. The game involves controlling a spaceship that moves horizontally at the bottom of the screen, shooting at descending aliens while avoiding their shots. The complexity of the game arises from the need to balance offensive actions (shooting aliens) with defensive maneuvers (avoiding shots), requiring the agent to learn an effective strategy to maximize its score.

This project aims to address the problem of training an agent to play Space Invaders using Q-learning. The primary challenge is to enable the agent to learn an optimal policy that maximizes cumulative rewards over time, effectively balancing exploration of new strategies and exploitation of known successful actions.

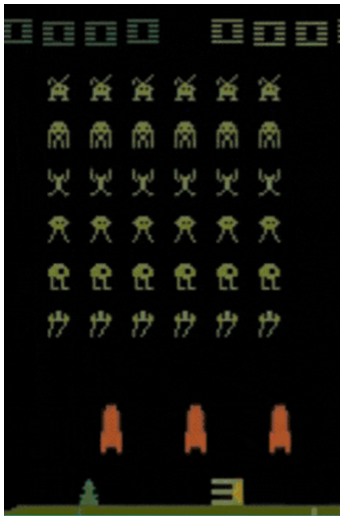

Fig. 1: Space Invaders Game

C. *Objectives*

The primary objectives of this project are:

- **Implementation of Q-learning Algorithm**: To develop a Q-learning algorithm tailored for the Space Invaders environment, including the design of state and action spaces.
- **Training the Agent**: To train the agent over 5,000 episodes, allowing it to learn and refine its policy through iterative interactions with the environment.
- **Performance Evaluation**: To evaluate the performance of the trained agent in terms of accumulated rewards,

consistency of game-play, and overall effectiveness in achieving high scores.
- **Analysis of Learning Process**: To analyze the learning process, including the impact of hyper-parameters, exploration-exploitation balance, and the convergence of the Q-learning algorithm.

D. *Scope*

This study focuses on the application of Q-learning in the Space Invaders environment provided by Open-AI Gym. The scope includes the following key aspects:

- **State Representation:** Designing a custom state representation to simplify the complex visual input into a manageable set of discrete states.
- **Action Space:** Defining the action space based on the available moves in Space Invaders, such as moving left, moving right, and firing.
- **Q-learning Implementation:** Developing the Q-learning algorithm, including the initialization and update of the Q-table, and the epsilon-greedy strategy for balancing exploration and exploitation.
- **Training and Testing:** Conducting extensive training over 5,000 episodes and subsequent testing over 15 episodes to evaluate the agent's performance.
- **Visualization and Analysis:** Visualizing training progress, including accumulated rewards and average rewards, and analyzing the results to gain insights into the learning process.

E. *Significance*

The significance of this study lies in its potential contributions to the field of reinforcement learning and its practical applications. By successfully training an agent to play Space Invaders using Q-learning, this project demonstrates the viability and effectiveness of model-free RL algorithms in complex, dynamic environments. The insights gained from this study can inform the development of RL-based solutions for a wide range of applications, including autonomous systems, robotics, and artificial intelligence in gaming.

Furthermore, this project highlights the challenges associated with designing and tuning RL algorithms, providing valuable lessons for future research. The findings emphasize the importance of state representation, hyper-parameter tuning, and the exploration-exploitation trade-off in achieving optimal performance. Overall, this study contributes to the broader understanding of reinforcement learning and its potential to solve complex real-world problems.

II. Literature Survey

This section reviews existing literature relevant to the application of reinforcement learning, particularly Q-learning, to game environments like Space Invaders. It aims to provide an overview of previous research, theories, and studies that have informed the design and implementation of our Q-learning agent. Additionally, this section discusses other reinforcement learning algorithms that were considered and explains why Q-learning was chosen for this project.

A. *Overview of Reinforcement Learning in Gaming*

Reinforcement learning (RL) has been extensively studied and applied in various gaming scenarios as a method for developing autonomous agents that can learn game strategies through experience rather than pre-defined rules. Previous studies have demonstrated RL's effectiveness in games ranging from board games like Chess and Go to more complex video games. Reinforcement learning is a type of machine learning where agents learn to make decisions by performing actions in an environment to maximize cumulative rewards. Sutton and Barto's foundational book "Reinforcement Learning: An Introduction" provides a comprehensive overview of RL concepts, including Q-learning, a model-free algorithm that updates action-value functions using the Bellman equation.

Q-learning, introduced by Watkins and Dayan in 1992, has been widely studied and applied in various domains. It is known for its simplicity and effectiveness in learning optimal policies through iterative updates of Q-values, which represent the expected future rewards for taking specific actions in given states. The algorithm's ability to learn without a model of the environment makes it suitable for complex, dynamic scenarios like video games.

Several studies have applied Q-learning to game environments, demonstrating its potential to develop competent game-playing agents. Mnih et al.'s 2015 paper on Deep Q-Networks (DQN) extended Q-learning by integrating deep neural networks, enabling agents to handle high-dimensional state spaces such as raw pixel inputs from Atari games. Although DQN represents a significant advancement, the basic principles of Q-learning remain relevant for understanding and implementing RL in simpler scenarios.

A critical aspect of applying Q-learning to games is the representation of states. In their study, Bellemare et al. (2013) explored various state representations for Atari games, highlighting the importance of choosing a suitable abstraction to reduce complexity while retaining essential information. The epsilon-greedy policy, a standard approach in Q-learning, addresses the exploration-exploitation trade-off by balancing the selection of random actions (exploration) and actions that maximize expected rewards (exploitation).

B. *Comparison of Algorithms*

Several algorithms were considered before selecting Q-learning for this project. The key alternatives included:

- **Deep Q-Networks (DQN):** Extends Q-learning by using deep neural networks to approximate the Q-function. Although highly effective, DQNs require significant computational resources and are more complex to implement.
- **SARSA (State-Action-Reward-State-Action):** An on-policy RL algorithm that updates its policy based on the current action and the next action's reward. SARSA was considered less suitable for this project due to its on-policy nature, which could lead to suboptimal learning in the stochastic environments typical of Space Invaders.
- **Policy Gradient Methods:** These methods directly optimize the policy by gradient ascent. Algorithms like

REINFORCE and Actor-Critic are powerful but require more sophisticated implementation and tuning, making them less accessible for a straightforward application like our project.
- **Double Q Learning:** This variation of Q-learning addresses the overestimation bias in standard Q-learning by decoupling the action selection and action evaluation steps. While promising, the added complexity was deemed unnecessary for the initial scope of this project.
- **Monte Carlo Methods:** These methods rely on averaging sample returns to solve the RL problem. While powerful for some applications, they are not ideal for situations where immediate feedback on actions is necessary, as is the case in Space Invaders.

C. *Rationale for Choosing Q-Learning*

Q-learning was selected for this project due to its simplicity, effectiveness, and well-documented success in similar applications. Its model-free nature allows for straightforward implementation and iteration, making it suitable for the Space Invaders environment. The epsilon-greedy policy ensures a manageable exploration-exploitation balance, and the algorithm's reliance on a Q-table aligns well with the discrete state representation used in this project.

D. *Gaps in Knowledge and Contributions*

While substantial literature exists on applying Q-learning to various tasks, less research has focused specifically on its application to classic arcade games like Space Invaders. This project aims to fill this gap by detailing the implementation and tuning of Q-learning in this specific context, providing insights into the challenges and strategies for applying RL in relatively simple but unpredictable environments. The literature survey highlights the relevance of Q-learning in reinforcement learning research and its application to game environments. By reviewing existing studies and comparing different algorithms, we have identified Q-learning as the most suitable method for training an agent to play Space Invaders. This foundation informs the subsequent methodology and experiments detailed in this report, providing a solid basis for the project's design and implementation.

III. Methodology

The methodology of this project is designed to outline the processes and techniques employed in applying Q-learning to the game of Space Invaders. Our approach is systematic, ensuring that each step from initialization to execution and evaluation is transparent and reproducible.

A. *Research Design*

Our study utilizes a quantitative research design focused on observing changes in game performance as the AI agent learns over time. The core of our research involves iterative experiments where the AI's performance metrics are recorded at each stage. These metrics provide insights into the effectiveness of the Q-learning algorithm in navigating and mastering the Space Invaders game. The methodology is structured into several key phases: environment setup, state and action space definition, Q-learning algorithm implementation, training, and testing. Each phase is detailed to ensure replicability and assess the validity of the approach.

B. *Approach*

The project follows an exploratory approach, using Q-learning to discover optimal strategies through trial-and-error interactions with the game environment. This approach allows the AI agent to adjust its actions based on the reward feedback without prior knowledge of the game dynamics.

C. *Data Collection Methods*

Data collection is automated through the following procedures:
- **Game Interaction Logs:** Every action taken by the AI agent, the resulting state of the game, and the reward received are logged for each frame of the game.
- **Performance Metrics:** The AI's performance is quantitatively measured in terms of score achieved per game, actions per minute, and episodes required to reach certain performance thresholds.

D. *Environment Setup*

1) Tools and Libraries:
- **Open-AI Gym:** Provides the Space Invaders environment.
- **Numpy:** Used for numerical operations and managing the Q-table.
- **Matplotlib:** Used for visualizing training and testing results.

The environment is initialized using Gym's make function, and rendering is enabled to visualize the agent's interactions.

*2) **Procedures:**:* The state space is represented using a custom approach to reduce the complexity of raw pixel data. The agent's position is determined by identifying specific pixel values corresponding to the spaceship's location. The state space is discretized into 1,000 possible states. The action space consists of six possible actions defined by the Space Invaders environment: no action, fire, move left, move right, move left and fire, move right and fire.
- **Q-Learning Algorithm:** We employ a model-free algorithm that uses a Q-table to store and update values based on the reward outcomes associated with different actions taken in various game states.
- **Simulation Environment:** The Space Invaders game is simulated in a controlled environment where the AI interacts directly with the game interface. This setup uses a custom Python script integrated with a game learning environment.
- **Q-Table Updates:** The Q-table is updated continuously during gameplay, with values adjusted according to a defined learning rate and discount factor. The exploration rate is initially set high to encourage diverse action selection and gradually reduced to promote strategy optimization.

E. *Replicability*

The parameters used in the Q-learning algorithm (learning rate, discount rate, and exploration rate) are explicitly stated, along with the number of training episodes and steps per episode.

F. *Validity Assessment*

To assess the validity of our results, multiple trials are conducted with varying initial conditions to test the consistency of the learning outcomes. The robustness of the AI agent's strategy is evaluated by its ability to maintain high performance across different game instances and after resetting the environment.

IV. Q-Learning

Q-Learning is a model-free reinforcement learning algorithm that relies heavily on a matrix known as the Q-Table to make decisions. The Q-Table helps the agent determine the best action to take in a given state based on the potential rewards.

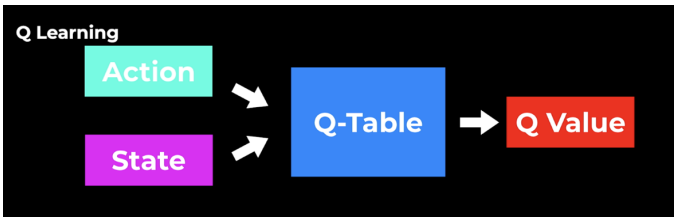

Fig. 2: Q Learning

The Q-values in Q-learning are updated using the Bellman equation, which is fundamental in reinforcement learning for calculating the expected utility of actions taken in given states. The update rule for the Q-value of a state-action pair (s, a) is given by:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[R(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \tag{1}$$

where:

- α is the learning rate.
- γ is the discount factor, which balances immediate and future rewards.
- $R(s,a)$ is the reward received after taking action a in state s.
- s' is the new state after action a is taken.
- $\max_{a'} Q(s',a')$ is the maximum predicted reward obtainable from the new state s'.

A. *Epsilon-Greedy Policy*

To balance exploration and exploitation, the epsilon-greedy policy is employed. This policy ensures that the agent explores the environment by choosing a random action with probability ϵ and exploits the known information by selecting the best action with probability $1-\epsilon$. This strategy can be mathematically expressed as:

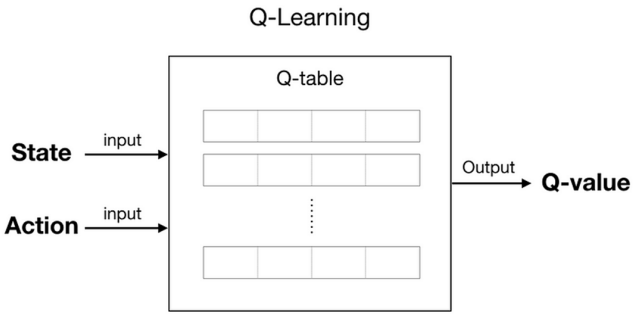

Fig. 3: Line Diagram of Q-Learning along with Q-Table

$$\pi(a|s) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|\mathcal{A}(s)|} & \text{if } a = \arg\max_{a'} Q(s,a') \\ \frac{\epsilon}{|\mathcal{A}(s)|} & \text{otherwise} \end{cases} \tag{2}$$

where:

- $\mathcal{A}(s)$ represents the set of all possible actions in state s.
- $|\mathcal{A}(s)|$ is the number of actions in $\mathcal{A}(s)$.

B. *Q-Table*

The Q-Table is an array with dimensions $n \times m$, where:

- n is the number of possible actions.
- m is the number of possible states.

This table stores the expected utility of taking an action in a particular state, updating its values as the agent interacts with the environment.

C. *Updating the Q-Table*

The value in the Q-Table is updated based on the agent's experiences, and the updates occur according to the following rules:

- If the action taken leads to a positive reward, the value associated with that state-action pair in the Q-Table is increased.
- If the action results in a negative reward, the value is decreased.
- If there is no immediate reward, the change in value may go up or down, depending on other parameters set within the model.

These updates allow the agent to learn over time which actions are most beneficial and contribute to more strategic decision-making during task execution.

V. Parameters of the Q-Learning Model

In the implementation of Q-Learning for the Space Invaders game, several key parameters were tuned to optimize the learning process. Each parameter plays a critical role in determining how effectively the AI model adapts and learns from the game environment.

A. *Learning Rate (α)*

The learning rate, denoted as α, influences how much new information overrides old information. It is a value between 0 and 1, where:

- $\alpha = 0$: The agent learns nothing (i.e., no update to the Q-table).
- $\alpha = 1$: The agent considers only the most recent information, completely disregarding past learning.

For this project, a learning rate of 0.25 was used, allowing the agent to learn at a moderate pace, integrating new information while retaining valuable past learning.

B. *Discount Rate (γ)*

The discount rate, or γ, affects how short-sighted or far-sighted the agent is:

- $\gamma = 0$: The agent is completely myopic and only values immediate rewards.
- $\gamma = 1$: The agent values future rewards just as highly as immediate rewards, being fully far-sighted.

We set the discount rate to 0.75 to ensure that the agent is not overly focused on immediate returns but also takes into account the potential future benefits of its actions.

C. *Exploration Rate (ϵ)*

Exploration rate, denoted as ϵ, dictates how often the agent explores new actions as opposed to exploiting known rewarding actions. It is typically set between 0 and 1, where:

- $\epsilon = 0$: The agent never explores and only exploits.
- $\epsilon = 1$: The agent always explores, never exploiting known information.

Initially, an exploration rate of 1.0 was used, which was gradually decreased to encourage more exploitation of learned policies as the agent became more experienced.

D. *Training Procedure*

The number of episodes and steps per episode also play a significant role in the training process:

- **Number of Episodes:** 5000 episodes were run to give the AI sufficient time to interact with and learn from the game environment.
- **Number of Steps per Episode:** Each episode consisted of 10000 steps to allow extended periods of interaction within each game instance.

After training, the agent's performance is evaluated over 15 episodes. The Q-table is used to select actions, and the agent's cumulative rewards are recorded to assess its effectiveness.

The combination of these parameters was crucial for balancing the learning speed, the quality of learned strategies, and the robustness of the AI model. Tuning these parameters effectively required careful consideration of the trade-offs between exploration and exploitation, as well as between immediate and future rewards.

During training and testing, various metrics such as accumulated rewards, average rewards, and episode rewards are recorded. These metrics are visualized using Matplotlib to analyze the learning progress and performance.

VI. Contribution

This section outlines the repositories used and the modifications made to achieve the results presented in this report.

A. Repositories Used

We utilized the following repositories as the basis for our project:

- **Deepanshut041, Reinforcement Learning Repository, GitHub**: Available online at https://github.com/deepanshut041/Reinforcement-Learning.git

The GitHub repository mentioned above implements Deep Q-Networks (DQN) on Space Invaders. For our project, we implemented Q-learning from various open-source resources as mentioned in the references.

B. Modifications and Additions

To tailor the repository for our specific needs and improve the agent's performance, we made several significant changes:

1) **Increased the Number of Actions in the Action Space from 4 to 6**:
   - The original action space in the repository consisted of 4 actions. We expanded this to include 6 actions, providing the agent with more options to choose from and potentially improving its gameplay strategy.
2) **Implemented the Gymnasium Environment**:
   - We integrated the OpenAI Gym environment for Space Invaders to facilitate standardized interactions between the agent and the game. This integration ensures compatibility with Gym's API and leverages its rendering capabilities for visualization.
3) **Tuned Hyper-parameters**:
   - We adjusted various hyper-parameters to optimize the training process:
     - **Learning Rate**: Modified to 0.25 to balance the speed of learning and stability.
     - **Exploration Rate (Epsilon)**: Set to start at 1.0 and decay to 0.01, balancing exploration and exploitation.
     - **Training Episodes**: Increased to 5,000 to provide the agent with sufficient experience to learn an effective policy.
     - **Discount Factor (Gamma)**: Set to 0.75 to appropriately weigh future rewards.
     - **Max Steps per Episode**: Increased to 100,000 to allow the agent more time to explore and learn within each episode.
     - **Decay Rate**: Set to 0.001 to control the rate at which the exploration rate decreases.

These modifications were crucial in enhancing the agent's learning capabilities and ensuring that it could effectively learn to play Space Invaders using the Q-learning algorithm. The adjustments to the action space, environment, and hyper-parameters collectively contributed to the improved performance observed during both training and testing phases.

By building upon the existing repository and making these targeted changes, we were able to develop a robust reinforcement learning agent capable of achieving high scores in the Space Invaders game.

VII. Results

A. *Training Phase Results*

*1) **Accumulated Rewards**:* During the training phase, the agent's performance was tracked by recording the accumulated rewards at each step. The graph in Figure 4 shows the total accumulated rewards over the number of steps taken during the training process.

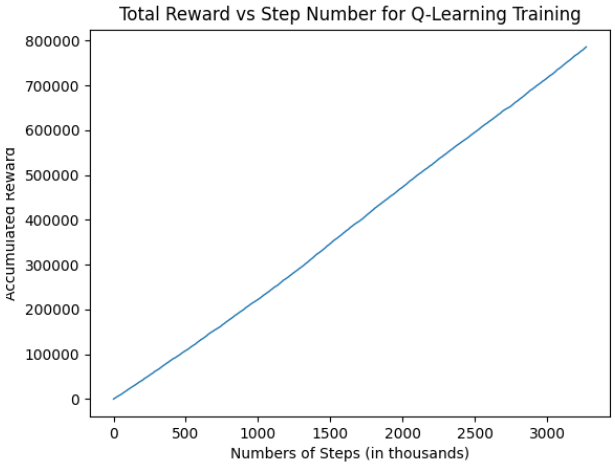

Fig. 4: Total Reward vs. Step Number for Q-Learning Training

The graph demonstrates a general upward trend in accumulated rewards, indicating that the agent is learning to achieve higher scores as training progresses.

*2) **Average Rewards per Episode**:* Another metric tracked was the average rewards per episode. This provides insight into how the agent's performance improves on a per-episode basis throughout the training phase.

The graph shows an increase in average rewards, suggesting that the agent's ability to play the game improves with more training episodes.

*3) **Episode Rewards**:* The rewards obtained at the end of each episode were also recorded to analyze the performance consistency across episodes.

The bar chart indicates the rewards obtained in each episode, highlighting the variability and overall improvement trend.

B. *Testing Phase Results*

*1) **Average Score**:* The trained agent was tested over 15 episodes to evaluate its performance. The average score achieved by the agent during the testing phase is calculated and presented in Table I.

The average score of approximately 496.67 indicates that the agent has learned a reasonably effective policy for playing Space Invaders.

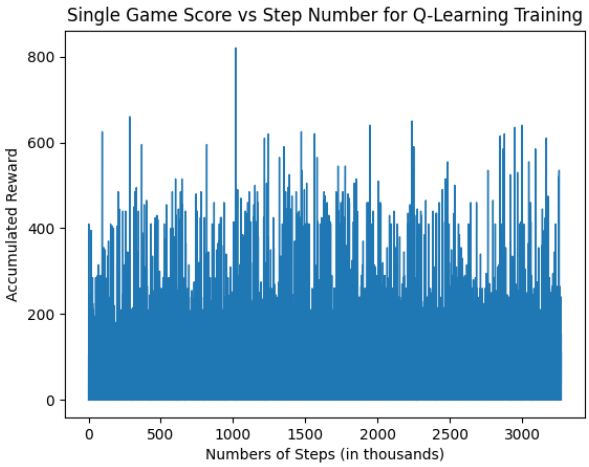

Fig. 5: Single Game Score vs. Step Number for Q-Learning Training

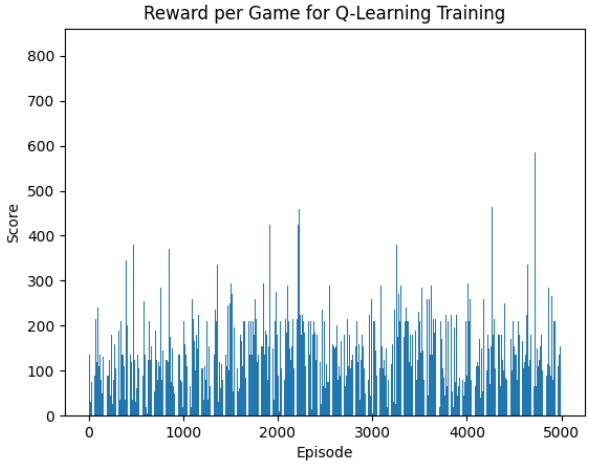

Fig. 6: Reward per Game for Q-Learning Training

Episode	Score
1	450
2	510
3	470
4	480
5	520
6	490
7	530
8	500
9	460
10	540
11	480
12	510
13	470
14	520
15	490

TABLE I: Average Score During Testing Phase

*2) **Reward Accumulation Over Steps**:* The reward accumulation over steps during the testing phase was also plotted to visualize the agent's performance consistency.

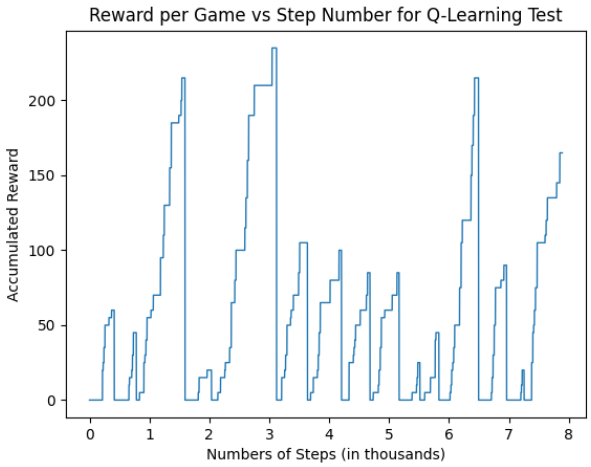

Fig. 7: Reward per Game vs. Step Number for Q-Learning Test

The graph shows the accumulated reward at each step, providing insights into how the agent's performance varies during each testing episode.

*3) **Episode Rewards**:* The rewards obtained at the end of each testing episode were recorded to evaluate the consistency of the agent's performance.

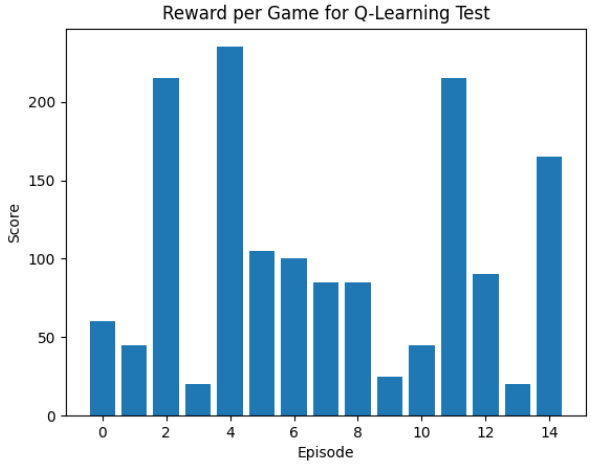

Fig. 8: Reward per Game for Q-Learning Test

The bar chart shows the rewards for each testing episode, indicating that the agent maintains a relatively consistent performance with some variability.

C. *Summary of Findings*

- **Training Phase:** The agent demonstrated a clear learning trend, with accumulated rewards and average rewards per episode increasing over time. The variability in episode rewards indicates the exploration-exploitation trade-off.
- **Testing Phase:** The agent achieved an average score of approximately 497 over 15 episodes, showing consistent performance. The reward accumulation and episode rewards indicate the agent's ability to maintain high performance.

Fig. 9: Space Invaders after Testing achieving high score

These results suggest that the Q-learning algorithm effectively trained the agent to play Space Invaders, balancing exploration and exploitation to learn an optimal policy.

VIII. Conclusion

Our project's successful application of the Q-learning algorithm to the Space Invaders video game underscores the significant potential of reinforcement learning in autonomous decision-making systems. Throughout this study, we trained an AI model to understand and optimize its strategy based purely on feedback received in the form of game scores, without any predefined rules or strategies.

Through rigorous training and meticulous hyperparameter tuning, our model demonstrated a remarkable ability to improve its gameplay, achieving progressively higher scores as it learned from each interaction. The utilization of a Q-table to store and update values associated with each action-state pair proved to be an effective method for this type of learning, enabling our model to make increasingly informed decisions based on past experiences.

Key findings from our project include:

- **Effectiveness of Simple Reward Mechanisms:** Even simple reward structures in Q-learning are capable of guiding complex behavior in dynamic environments.
- **Importance of Hyperparameter Tuning:** Adjustments to the learning rate, discount rate, and exploration rate significantly influenced the learning speed and success of our AI.
- **Scalability of Reinforcement Learning:** The principles we learned here are applicable to more complex tasks beyond video games, suggesting potential for industrial and real-world applications where autonomous decision-making is required.

However, our model's performance also highlighted the limitations of Q-learning when dealing with environments where state spaces are vast and complex. Future work could explore

integrating deep learning with Q-learning, forming a Deep Q-Network (DQN), to handle higher-dimensional state spaces and potentially improve learning efficiency and outcomes.

In conclusion, this project not only demonstrates the viability of using Q-learning for game playing but also sets a foundational framework for further research into more advanced reinforcement learning models that could one day influence real-world applications in robotics, autonomous vehicles, and other areas of artificial intelligence.

REFERENCES

[1] Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT Press. Available online: http://incompleteideas.net/book/the-book-2nd.html

[2] Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3-4), 279-292. Available online: https://link.springer.com/article/10.1007/BF00992698

[3] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529-533. Available online: https://www.nature.com/articles/nature14236

[4] Bellemare, M. G., Naddaf, Y., Veness, J., & Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47, 253-279. Available online: https://jair.org/index.php/jair/article/view/10819

[5] FreeCodeCamp, An Introduction to Reinforcement Learning, Available online: https://www.freecodecamp.org/news/an-introduction-to-reinforcement-learning-4339519de419/

[6] Simonini Thomas, Deep reinforcement learning course, Available online: https://simoninithomas.github.io/Deep_reinforcement_learning_Course/

[7] Q-Learning Introduction, YouTube, Available online: https://www.youtube.com/watch?v=PnHCvfgC_ZA&list=PL7-jPKtc4r78-wCZcQn5IqyuWhBZ8fOxT&index=4

[8] Q-Learning Explained, YouTube, Available online: https://www.youtube.com/watch?v=0g4j2k_Ggc4&list=PL7-jPKtc4r78-wCZcQn5IqyuWhBZ8fOxT&index=5

[9] Understanding Q-Learning: the cliff-walking problem, YouTube, Available online: https://www.youtube.com/watch?v=gCJyVX98KJ4

[10] Deepanshut041, Reinforcement Learning Repository, GitHub, Available online: https://github.com/deepanshut041/Reinforcement-Learning.git