**National University of Computer & Emerging Sciences**

**Karachi Campus**



**AI Snake Game**

**Submitted By:**

K21-3216 Zaki Mustafa  
k22-4799 Rohaan Ehtesham

**Course:**  
Artificial intelligence (AI)

**Instructor:**  
Ms Alishba Subhani

**Submission Date:**  
5/11/2025

**1. Executive Summary**

**Project Overview:**  
This project aimed to implement an AI-powered version of the classic Snake game using a reinforcement learning technique called Deep Q-Learning (DQN). The objective was to enable an AI agent to learn optimal strategies for playing Snake autonomously by interacting with the environment, receiving feedback in the form of rewards, and updating its policy over time. Modifications were made to the conventional game to support AI integration and facilitate real-time learning and decision-making.

**2. Introduction**

**Background:**  
The Snake game is a classic arcade game in which the player controls a snake to eat food and grow longer, while avoiding collisions with walls and itself. Due to its simple rules and grid-based environment, it is ideal for experimenting with AI techniques. This project selects Snake for its compatibility with reinforcement learning methods and modifiability for continuous training scenarios.

**Objectives of the Project:**

* Develop a Snake game environment suitable for AI training.
* Implement a DQN-based AI agent that can learn to play the game.
* Train the agent through trial-and-error interaction and reward feedback.
* Evaluate the agent’s performance over time.

**3. Game Description**

**Original Game Rules:**  
In the traditional Snake game, the snake moves continuously in one of four directions (up, down, left, right). The player must control the snake to collect food, which increases its length and score. The game ends when the snake hits the wall or itself.

**Innovations and Modifications:**

* Developed a fully autonomous AI player using DQN.
* Enabled continuous gameplay for long-term training.
* Introduced a frame-based penalty system to prevent aimless movement.
* Visualized scores and average performance using a live plot.

**4. AI Approach and Methodology**

**AI Techniques Used:**

* Reinforcement Learning (specifically Deep Q-Learning).
* Experience Replay to stabilize learning.
* Epsilon-Greedy strategy for exploration vs. exploitation.
* PyTorch-based neural network for function approximation.

**Algorithm and Heuristic Design:**

* The agent’s state consists of 11 binary features, including danger indicators, current direction, and food location.
* The neural network architecture: input (11 nodes) → hidden (256 nodes) → output (3 nodes corresponding to move straight, turn right, turn left).
* Reward system:
  + +10 for eating food.
  + -10 for collisions.
  + 0 for each move unless a terminal state is reached.

**AI Performance Evaluation:**

* Performance metrics: score, mean score, and record score over games.
* Visual feedback via a live performance graph.
* Decision-making efficiency inferred from training logs and performance trends.

**5. Game Mechanics and Rules**

**Modified Game Rules:**

* The snake receives a reward/penalty after each action based on outcomes (eat, die, survive).
* AI-only gameplay with automatic reset after each game-over condition.

**Turn-based Mechanics:**

* Each frame represents a decision turn for the AI.
* Game ends when the snake hits a wall, itself, or moves without progress for too long.

**Winning Conditions:**

* No explicit “win” state; goal is to maximize score and longevity.

**6. Implementation and Development**

**Development Process:**

* Built the game environment using Pygame.
* Designed and trained the AI agent using PyTorch.
* Separated modules into agent.py, snake\_game.py, model.py, and plotter.py for maintainability.

**Programming Languages and Tools:**

* **Programming Language:** Python
* **Libraries:** Pygame, PyTorch, NumPy
* **Tools:** GitHub for version control

**Challenges Encountered:**

* Balancing exploration and exploitation in the early training stages.
* Ensuring stable training via memory replay and batch processing.
* Rendering real-time graphics while training the agent without lags.

**7. Team Contributions**

**Team Members and Responsibilities:**

* **Zaki Mustafa:** Developed and optimized agent.py,model.py and plotter.py. Modified game rules and environment for reinforcement learning compatibility.
* **Rohaan Ehtesham:** Developed and optimize snake\_game.py. Conducted experiments and performance evaluations.

**8. Results and Discussion**

**AI Performance:**

* The AI improved consistently, with scores increasing after each training cycle.
* Achieved an average decision time of less than 0.1 seconds per frame.
* Reached a record score of [record value] after [number of games] episodes.
* Demonstrated the capability to survive longer and consistently seek food over time.

**9. References**

* Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction*.
* PyTorch Documentation – <https://pytorch.org>
* Pygame Documentation – https://www.pygame.org/docs/
* GitHub repositories and tutorials on DQN Snake implementations