**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. Project Overview**

**Project Topic:**  
This project is based on the classic Snake arcade game, which has been transformed into a single-player training environment where an AI agent learns to play using Reinforcement Learning. The conventional Snake gameplay is modified to support an autonomous agent powered by Deep Q-Learning (DQN), allowing it to improve its performance through self-play.

**Objective:**  
The main objective of this project is to develop a self-learning AI agent capable of playing Snake efficiently. The project uses a Deep Q-Network to teach the agent optimal strategies through rewards and penalties, simulating a reinforcement learning environment with real-time feedback and adaptive decision-making.

**2. Game Description**

**Original Game Background:**  
Snake is a simple single-player arcade game in which a player controls a snake that grows in length by eating food items while avoiding collisions with walls or itself. The game ends when the snake crashes into the wall or its own body.

**Innovations Introduced:**

* The game is modified to support **autonomous AI play**, replacing human controls.
* A **reward system** is implemented: +10 for food, -10 for collision, and 0 for a step.
* **Continuous play** is added to allow long-term training with automatic resets.
* **Real-time performance tracking** is introduced via score graphs.

These enhancements allow the Snake environment to serve as a reinforcement learning testbed with increasing gameplay complexity based on the agent's experience.

**3. AI Approach and Methodology**

**AI Techniques to be Used:**

* **Deep Q-Learning (DQN):** A neural network-based Q-learning method where the model predicts Q-values for each possible action given a state.
* **Experience Replay:** The agent stores past experiences and samples random batches to break correlation between consecutive states.
* **Epsilon-Greedy Policy:** A balance between exploration (random moves) and exploitation (using learned moves).

**Heuristic Design:**

* The state includes 11 binary features such as obstacle danger in each direction, the snake’s current movement direction, and the food's relative location.
* The reward function incentivizes food collection and penalizes collisions to guide learning.

**Complexity Analysis:**

* The time complexity per training step depends on the neural network's forward pass: O(n) where n is the number of input features.
* Memory usage grows with experience replay buffer size (maxlen = 100,000) and batch size (1,000).
* Complexity is manageable due to simple game dynamics and low-dimensional state space.

**4. Game Rules and Mechanics**

**Modified Rules:**

* Player input is removed and replaced by AI decision-making.
* The game ends if the snake hits a wall, itself, or exceeds a step limit without progress.
* A food item is randomly placed after each collection.

**Winning Conditions:**

* No definitive win state; the agent aims to survive and maximize the score.
* The longer the snake survives and the more food it eats, the better its performance.

**Turn Sequence:**

* Each game frame acts as one turn.
* The AI selects one of three possible moves: straight, turn left, or turn right.
* The game resets automatically upon snake death, beginning a new training episode.

**5. Implementation Plan**

**Programming Language:**  
Python

**Libraries and Tools:**

* **Pygame:** For GUI and game rendering
* **PyTorch:** For implementing the DQN agent and training
* **NumPy:** For array and mathematical operations
* **Matplotlib:** For performance plotting

**Milestones and Timeline:**

* **Week 1–2:** Setup game environment and define AI state representation
* **Week 3–4:** Implement the DQN model, training loop, and reward system
* **Week 5–6:** Integrate AI with Snake game and test short-term memory learning
* **Week 7:** Optimize training using experience replay and plot performance
* **Week 8:** Final performance testing and prepare project report

**6. References**

* Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction*
* PyTorch Documentation – <https://pytorch.org>
* Pygame Documentation – https://www.pygame.org/docs/
* Code tutorials on Deep Q-Learning for Snake (e.g., @python-engineer on YouTube)
* GitHub repositories for DQN Snake implementations