AMRITA VISHWA VIDYAPEETHAM

22AIE201 : Fundamentals of AI

An AI Agent for the Snake Game Using Reinforcement Learning

Team 7 :
G Prajwal Priyadarshan - 214
Kabilan K - 224
Kishore B - 227
Rahul L S - 248

Introduction

Artificial Intelligence (AI) has made significant advances, particularly in **Reinforcement Learning (RL)**, which enables agents to make strategic decisions by interacting with environments. The **Snake game**, a well-known arcade game, serves as an excellent platform to test RL agents. In this project, we explore how an AI agent can learn and optimize its actions in the Snake game using **Q-Learning**, a core RL algorithm.

- **Reinforcement Learning (RL)** is a branch of machine learning where an agent learns by interacting with an environment.
- **Q-Learning** is a model-free RL algorithm that teaches the agent which action to take to maximize rewards.
- Here model-free means the agent doesn't know what will happen if it moves right or left, it just tries and learns what works based on rewards (like getting food or hitting a wall).

Motivation

This project presents the design and implementation of an intelligent agent that learns to play the Snake game using **Q-Learning**, a type of Reinforcement Learning (RL). The agent is trained to navigate a grid-based environment, collect food, and avoid self-collisions and walls. The game is built using Python's **Pygame**, and the learning process is driven by rewards and penalties based on actions taken. This hands-on approach provides valuable insights into training AI agents, optimizing policies, and applying **Q-Learning/DQN** techniques to dynamic and strategic environments.

Base Paper - Developing Game AI for Classic Games Using Reinforcement Learning

For Snake (Q-learning)

- **Environment Setup**:
  - **States**: Snake’s position, food location, and obstacles (walls or itself).
  - **Actions**: Four directions – up, down, left, right.
- **Q-Learning Algorithm**:
  - Maintains a **Q-table** to store expected rewards for each state-action pair.
  - Uses **ε-greedy strategy** for balancing exploration and exploitation.
  - Applies **experience replay** to reuse past experiences for better learning.

🐍 **Snake Game Agent**:

- Learns by **interacting with the grid environment**.
- Gets **positive rewards** for eating food and **negative rewards** for crashing.

Literature Survey

Name	Author Name	Methods
Developing Game AI for Classic Games Using Reinforcement Learning	Shivam Gupta, 2024	Q - Learning
Deep Q-Snake: An Intelligent Agent Mastering the Snake Game with Deep Reinforcement Learning	Debjyoti Ray, 2024	DQN [Deep Q Learning]
A Deep Q-Learning based approach applied to the Snake game	Alessandro Sebastianelli, 2021	DQN [Deep Q Learning]
Training a Reinforcement Learning Agent based on XCS in a Competitive Snake Environment	Johannes B¨ uttner, 2021	XCS [Extended Classifier System]

Research Objectives

- ❑ To design an environment-agent interaction model for the Snake game
- ❑ To implement Reinforcement Learning for training the agent
- ❑ To develop a suitable reward mechanism that drives optimal learning
- ❑ To train the agent through repeated trials (episodes) and analyze learning performance
- ❑ To demonstrate path optimization and intelligent decision-making in gameplay
- ❑ To explore generalization and limitations of RL in real-time environments
- ❑ To compare performance of Value based Reinforcement Learning types

Methodology

Define the Snake Game Environment

State: What the agent observes at each step.

- Grid size
- Snake's head position
- Food position
- Direction of movement
- Obstacles (walls, body)

Action Space:

- 3 discrete actions: [turn left , go straight, Turn Right] relative to the current direction.

Reward Function (crucial for learning):

- +10 for eating food
- -100 for dying
- -0.1 small penalty for every step to Encourage shorter paths

Build the Q-learning Agent

Create a table (called Q-table)

- This table helps the agent "remember" what to do in different situations.
- Rows = situations (states), Columns = actions (turn left, go straight, turn right)

Teach the agent by giving rewards

- After each move, the agent gets feedback (reward).
- If the move was good (ate food) → reward is high ,If the move was bad (hit wall) → penalty is high
- This reward helps the agent learn which actions are better over time.

Balance learning and trying

- In the beginning, the agent tries random moves to explore the game.
- Slowly, it starts choosing smarter moves it has learned (exploiting knowledge).

Repeat the process

- The agent plays the game many times. Each time, it learns more from its mistakes and improves.

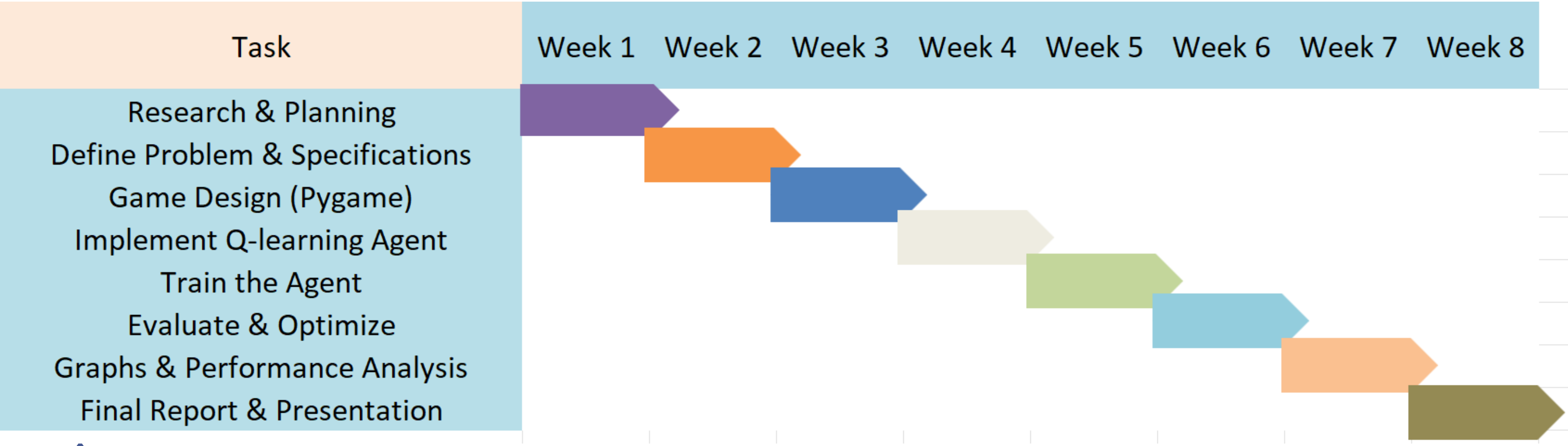
Train the Agent

- Let the agent play, firstly it explores, and overtime it starts to make smarter moves, this process is repeated until the agent improves

Test and Evaluate the Agent

- After training, we test how well the agent plays.
- Record results Show graphs or visuals (like score vs number of games) to prove the agent learned.

Gantt Chart


Task
Week 1
Week 2
Week 3
Week 4
Week 5
Week 6
Week 7
Week 8
Research & Planning
Define Problem & Specifications
Game Design (Pygame)
Implement Q-learning Agent
Train the Agent
Evaluate & Optimize
Graphs & Performance Analysis
Final Report & Presentation

Future Scope

- Can be extended to play more **complex games** like Pac-Man or Mario.
- Use **better RL algorithms** (Double DQN, PPO, Actor-Critic) for faster and smarter learning.
- Try **multi-snake games** (competition or teamwork).
- Use **step-by-step training** (start with easy levels, then harder).
- Make the AI more **explainable** so humans can understand its moves.
- Apply the same idea to **real-world problems** like robot path planning, self-driving cars, or network optimization.
- Create **human-friendly agents** that learn and adapt to human players.

Conclusion

This project aims to develop an AI agent capable of playing the classic Snake game using value based reinforcement learning techniques By interacting with its environment and receiving feedback in the form of rewards and penalties, the agent learns optimal strategies for survival and food collection over time.

The game environment is modeled as a grid, and the agent continuously improves its decision-making by updating its action values or training a neural network. This project effectively demonstrates Fundamental AI concepts like environment-agent interaction, reward-based learning, exploration vs. exploitation, and function approximation, offering a strong foundation in practical reinforcement learning applications.

Thank You