University of New Haven

Tagliatela College of Engineering

TERM PROJECT - AI-Powered Snake Game using Reinforcement Learning

Harinath Talluri(00865329)

Abhinay Reddy Musuku(00858095)

Madhava Naidu Valluru(00884651)

Introduction to Artificial Intelligence

Under the Supervision of

Dr. Shivanjali Khare

1. Research Question

This project aims to develop a Snake AI agent that uses reinforcement learning algorithms, specifically Q-learning and Deep Q-Networks (DQN), to maximize its score and minimize collisions, thereby improving performance and survival.

---

2. Introduction

The Snake game is a dynamic environment where the player controls a snake to eat food and avoid collisions. This project aims to develop an AI agent using Reinforcement Learning (RL), specifically Q-learning and DQN, to autonomously play and maximize its score. Python and PyGame are used for the game environment, PyTorch for RL model implementation, and Matplotlib for performance visualization. Challenges include the agent getting stuck in loops, trapping itself, and optimizing training parameters for efficiency.

---

3. Objectives

Primary Objectives:

- **Teach AI to Play Snake**: Enable the AI to learn gameplay strategies using reinforcement learning.
- **Maximize Score**: Develop intelligent pathfinding strategies to efficiently collect food.
- **Ensure Survival**: Minimize the likelihood of collisions with itself and the game boundaries.

Secondary Objectives:

- **Understand Reinforcement Learning**: Gain practical knowledge of RL concepts.
- **Enhance Problem-Solving Skills**: Address challenges related to game state representation, reward systems, and model training.

---

4. Related Literature

1. **Deep Q-Learning for Games** (Mnih et al., 2015): Demonstrates the application of DQN to Atari games, showing how RL paired with neural networks can learn optimal policies for complex environments.
2. **Snake AI using RL**: Explores Q-learning and reward-based reinforcement learning to enable agents to navigate simple environments like Snake.

---

5. Methodology

Tools:

- **Python**: Main programming language for implementation.
- **PyGame**: Used to create the game environment, handle graphics, and manage game interactions.
- **PyTorch**: Implements reinforcement learning algorithms and neural network training.
- **Matplotlib**: Visualizes training progress by graphing metrics like scores and cumulative rewards.
- **GitHub**: Maintains version control and documents progress.

Reinforcement Learning Approach

- **Q-Learning**: Q-learning is a model-free RL algorithm that updates the Q-value of a state-action pair (s,a)(s, a)(s,a) based on the Bellman equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right)$$

- **s**: Current state
- **a**: Action taken
- **r**: Reward received
- **s′**: Next state
- **α\alphaα**: Learning rate
- **γ\gammaγ**: Discount factor

```
# 1: predicted Q values with current state
pred = self.model(state)

target = pred.clone()
for idx in range(len(done)):
    Q_new = reward[idx]
    if not done[idx]:
        Q_new = reward[idx] + self.gamma * torch.max(self.model(next_state[idx]))

    target[idx][torch.argmax(action[idx]).item()] = Q_new
```

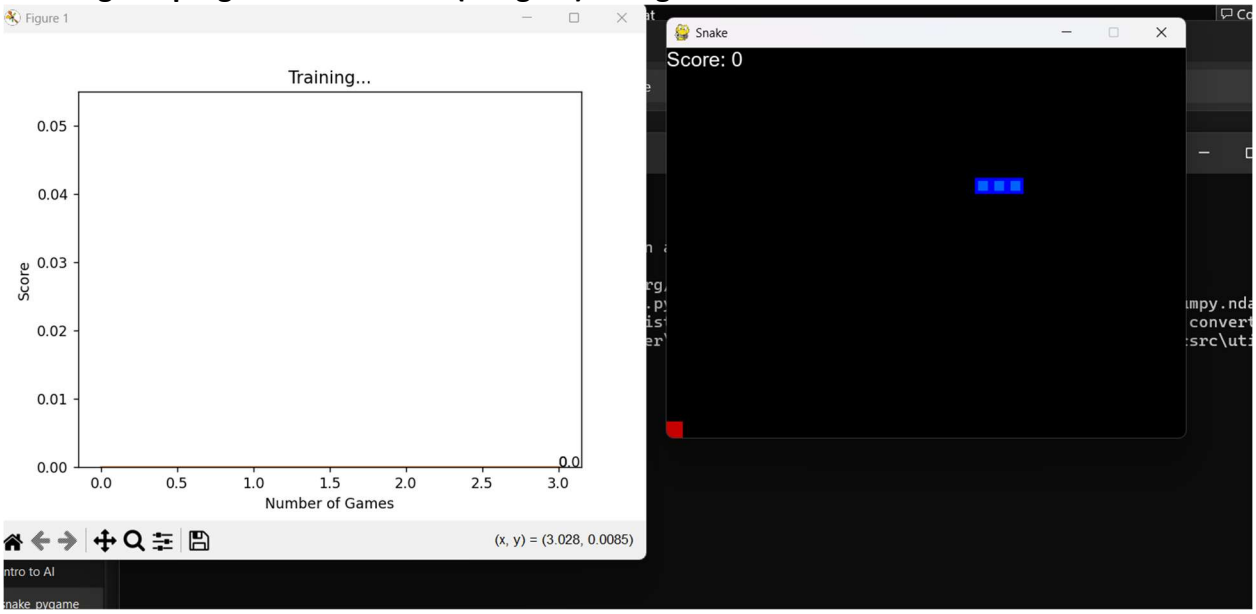
DQN Training Algorithm:

- Predict Q-values for the current state using the **main network**.
- Compute the target Q-values using the **target network**

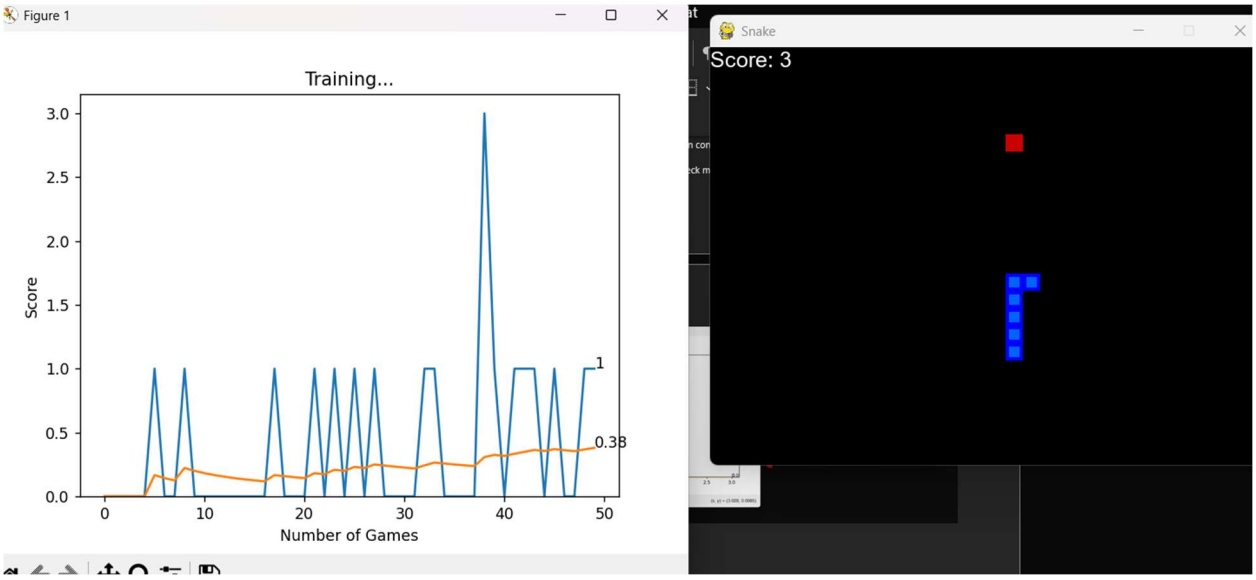
$$Q_{\text{target}} = r + \gamma \max_{a'} Q_{\text{target}}(s',a')$$

```
class QTrainer:
    def __init__(self, model, lr, gamma):
        self.lr = lr
        self.model = model
        self.gamma = gamma
        self.optimizer = optim.Adam(model.parameters(), lr=self.lr)
        self.criterion = nn.MSELoss()
```

Tracking the progress of the snake(AI agent) First game



After 1 minute



After 2 minutes