Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

A Project Report
on
"PySerpent: AI-Powered Snake Mastery "

[Code No:AIPC 201]

(For partial fulfillment of 2nd Year/ 2nd Semester in Computer Science/Engineering)

Submitted by

"Aman Kumar Jha (12)
Prasiddha Koirala(16)
Pratiksha Mishra (19)"

Submitted to

Mr. Sunil Regmi

Department of Computer Science and Engineering

Submission Date:01/01/2024

Table of Contents

ACKNOWLEDGEMENT

We would like to extend our heartfelt appreciation to Mr.Sunil Regmi for his unwavering support, guidance, and invaluable feedback throughout the development of this project. His expertise and encouragement have been pivotal in shaping the direction and success of our endeavor.

We are immensely grateful to each member of our team for their dedication, hard work, and collaboration. This project wouldn’t have been possible without the collective effort, insightful discussions, and mutual support among us.

We also express our gratitude to free online journals for providing essential tools, frameworks, and resources that significantly contributed to the project's realization.

Furthermore, we wish to thank our families and friends for their patience, understanding, and encouragement during the course of this project. Their support has been a constant source of motivation for us.

Yours Sincerely,

Aman Kumar Jha

Prasiddha Koirala

Pratiksha Mishra

Abstract

This project explores the convergence of Python programming, PyTorch, Pygame, and reinforcement learning techniques to develop an artificial intelligence (AI) agent capable of mastering the classic Snake game. Leveraging the powerful neural network capabilities offered by PyTorch, combined with the interactive gaming environment of Pygame, our objective is to train an AI agent to autonomously navigate the Snake game grid, maximizing its score through strategic decision-making.

The project entails the construction of a game environment using Pygame, where the Snake game unfolds within a defined grid, presenting challenges and rewards. The AI agent, implemented using PyTorch, undergoes reinforcement learning to decipher optimal actions based on the game state, aiming to achieve prolonged survival and maximize its score. This involves training the AI to learn patterns, make informed decisions, and adapt its strategies dynamically.

Through iterations of reinforcement learning algorithms and neural network optimization, the AI agent progressively refines its gameplay strategy, demonstrating adaptive behaviors, and enhancing its ability to navigate the game environment more efficiently.

The culmination of this project not only showcases the integration of Python-based technologies but also presents a functional AI-driven solution capable of excelling at the Snake game, illustrating the potential of reinforcement learning in gaming environments.

Keywords:Python, PyTorch, Pygame, Reinforcement Learning, Artificial Intelligence, Neural Networks, Game Development, Snake Game, Training AI Agents, Machine Learning, Deep Q-Networks, Game Environments, Strategy Learning, Autonomous Agents, Optimization Techniques.

Acronyms/Abbreviations (if any)

The list of all abbreviations used in the documentation is included in this section. See the example below

RL:	Reinforcement Learning
NN:	Neural Networks
DQN:	Deep Q-Networks
AA:	Autonomous Agents
OT:	Optimization Techniques
GD:	Game Development
SL:	Strategy Learning

Chapter 1 Introduction

1.1 Background

In the sphere of AI and gaming, the fusion of Python, PyTorch, Pygame, and reinforcement learning defines a groundbreaking quest: training an AI to excel at the classic Snake game. Leveraging PyTorch's neural network capabilities and Pygame's interactive playground, this project embarks on equipping an AI agent with the finesse to maneuver the grid, strategically gather rewards, and expertly evade obstacles. Through iterative learning empowered by reinforcement learning techniques, the AI undergoes a process of trial and adaptation, gradually mastering the nuances of Snake. PyTorch's computational might drives the AI's decision-making while Pygame provides the visual canvas for real-time interaction. The ultimate aim is to witness the AI agent evolve, learning to navigate the ever-changing terrain, demonstrating prowess, and achieving mastery in the captivating challenge of the Snake game.

1.2 Objectives

- To implement a reinforcement learning (RL) system to train an artificial intelligence (AI) agent to play the classic Snake game.

1.3 Motivation and Significance

This project's motivation stems from the intersecting domains of AI, gaming, and technological innovation. By combining Python's versatility, PyTorch's robust neural network capabilities, Pygame's interactive environment, and reinforcement learning techniques, we aim to create an AI agent capable of mastering the Snake game. The significance lies in showcasing the practical implications of these technologies, demonstrating how reinforcement learning empowers AI agents to learn, strategize, and excel in complex gaming environments. This project not only underscores the synergy between programming, neural networks, and gaming frameworks but also highlights the potential for AI-driven solutions in mastering intricate tasks within gaming scenarios, offering insights into broader applications of AI in problem-solving and decision-making contexts. Ultimately, this endeavor seeks to illustrate the transformative potential of integrating these technologies while fostering a deeper understanding of their capabilities in AI-driven gameplay mastery.

Chapter 2 Methodology

The methodology for the project of training an AI to play Snake using Python, PyTorch, and Pygame involves several key steps. Below is a high-level outline of the methodology:

Problem Definition:

- Clearly define the problem you are trying to solve: training an AI agent to play the Snake game using reinforcement learning.

Environment Setup:

- Set up your development environment with the necessary tools and libraries, including Python, PyTorch, and Pygame.

Game Environment Implementation:

- Implement the Snake game environment using Pygame. Define the rules, mechanics, and visual representation of the game.

State Representation:

- Decide on the state representation for the Snake game. Identify the information the AI agent needs to make decisions, such as the positions of the snake, the location of food, and the game board.

Action Space:

- Define the action space for the AI agent, specifying the possible moves it can make in response to the observed state.

Neural Network Architecture:

- Design the neural network architecture using PyTorch. The neural network should take the game state as input and output the corresponding action probabilities.

Reinforcement Learning Algorithm:

- Choose a reinforcement learning algorithm, such as Q-learning, deep Q-networks (DQN), or policy gradient methods. Implement the selected algorithm to train the AI agent.

Reward Design:

- Design a reward function that provides feedback to the AI agent. Define positive and negative rewards based on the agent's actions and performance in the game.

Training:

- Train the AI agent by repeatedly playing the Snake game and updating the neural network weights to maximize cumulative rewards. Monitor the training process and adjust hyperparameters as needed.

Evaluation:

- Evaluate the performance of the trained AI agent on the Snake game. Measure key metrics such as average score, length of survival, and the agent's ability to avoid collisions.

Chapter 3 Design and Implementation

3.1 System Requirements Specifications

3.1.1 Hardware Requirements

Minimum System Configuration:

- Processor: Specify processor requirements (e.g., Intel Core i5 or equivalent)
- Memory: Minimum RAM required (e.g., 8GB)
- Storage: Required disk space (e.g., 256GB SSD)

3.2.2 Software Requirements

Following are the software that were used during the completion of the project:

- **Python:** Python is the primary programming language for this project.
- **Pytorch:** Deep learning framework used to implement and train your neural network for reinforcement learning.
- **Pygame:** Pygame is a set of Python modules designed for writing our snake game.
- **Numpy:**NumPy is a fundamental package for scientific computing with Python.

Chapter 4 Discussion on the achievements

4.1 Challenges Faced

Complexity of the Snake Game:

- Depending on the complexity of your Snake game implementation, the AI might need to learn intricate patterns and strategies to maximize its score.

Reinforcement Learning Complexity:

- Reinforcement learning, especially when working with neural networks, can be complex. Fine-tuning hyperparameters, choosing appropriate reward structures, and dealing with exploration-exploitation trade-offs can be challenging.

State Representation:

- Designing an effective state representation for the Snake game can be crucial. Deciding what information the AI needs to make decisions is a non-trivial task.

Reward Design:

- Designing a reward function that encourages the AI to behave optimally can be challenging. Improperly designed rewards may lead to suboptimal or even counterintuitive behaviors.

Chapter 5 Conclusion and Recommendation

5.1 Limitations:

While the project of training an AI to play Snake using Python, PyTorch, and Pygame is a valuable learning experience, it also comes with some limitations. Here are some potential limitations of this project:

Simplicity of the Snake Game:

- The Snake game is relatively simple compared to more complex video games. While this simplicity may make it a good starting point for reinforcement learning projects, it might not capture the challenges presented by more intricate games or real-world applications.

Limited Generalization:

- The AI agent trained on the Snake game may become proficient at playing the game but might not generalize well to different environments or games.

Small State Space:

- The state space of the Snake game is limited, and the action space is discrete. In more complex environments, dealing with continuous state and action spaces becomes a more challenging problem.

Reward Sparsity:

- The reward signal in the Snake game can be sparse. The agent may receive positive rewards only when it eats the food or negative rewards when it collides with the wall or itself. Sparse rewards can make learning more difficult for the AI.

Limited Real-world Applications:

- While training an AI to play games is interesting, the direct applicability of the learned policies to real-world problems might be limited. Real-world problems often involve more uncertainty and complexity.

5.2 Future Enhancements:

Human-AI Collaboration:

- Explore ways to allow human players to collaborate with the AI agent. This could involve a co-play mode where the AI assists the human player or a mode where the AI learns from observing human gameplay.

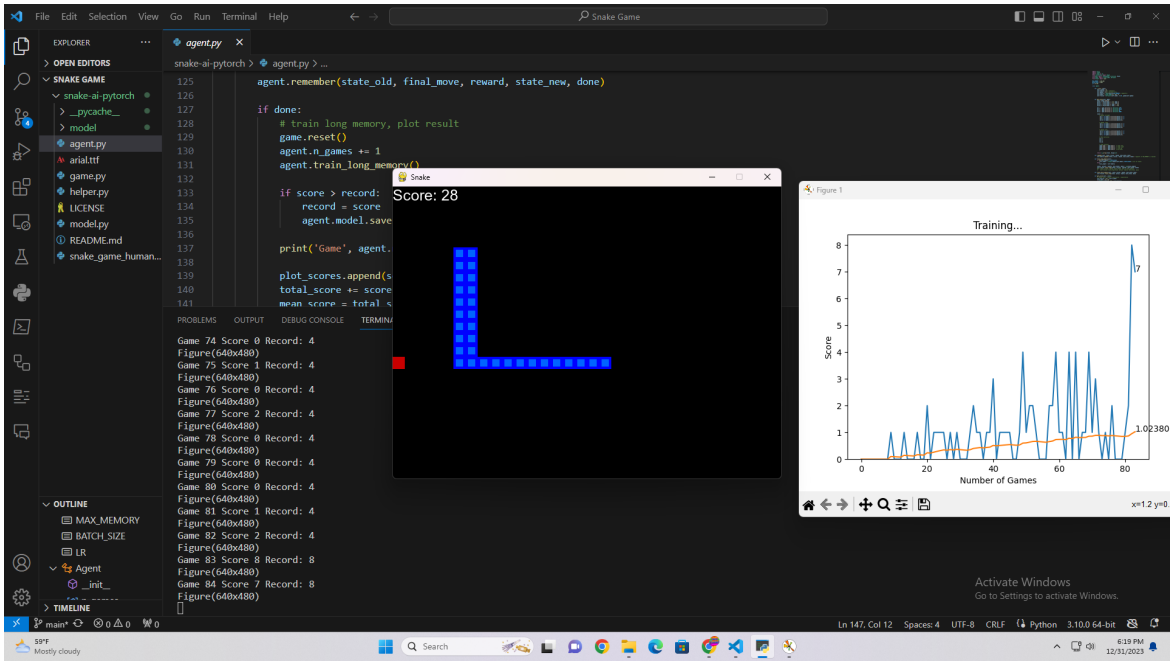
Real-time Learning:

- Implement a system that allows the AI agent to learn in real-time as the game is being played. This can involve continuously updating the agent's policy based on new experiences.

Integration with Cloud Services:

- Explore the integration of the project with cloud-based services for distributed training or for deploying and testing the AI agent on remote servers.

APPENDIX:


Snake
Score: 28
Training...
Number of Games
Score
Activate Windows
Go to Settings to activate Windows.