DEEP LEARNING BASED AI MODEL FOR SOLVING GOBBLET BOARD GAME

By : JK

ABSTRACT

The use of deep reinforcement learning techniques to master the game of Gobblet. Gobblet is a strategic board game played on a 4x4 board with twelve pieces of varying sizes per player. The objective is to form a row of four pieces, horizontally, vertically, or diagonally. To propose and implement a deep reinforcement learning approach that models the game as a Markov Decision Process (MDP), with states representing board positions, actions representing piece placement or movement, and rewards indicating game outcomes. The method employs a deep Q-network (DQN) to approximate the state-action value function and is trained using experience replay and target networks for improved stability. Through analysis of the learned policy, insights into the agent's strategy is inferred. It is observed that the agent learns to prioritize protecting its own pieces while simultaneously obstructing its opponent's pieces. These results highlight the effectiveness of deep reinforcement learning for achieving high-level gameplay in Gobblet. This approach has the potential to be extended to other strategy board games and can contribute to the development of advanced game-playing agents.

CONTENTS

List of Figures

List of Abbreviations

DQN	Deep Q Networks
PDF	Portable Document Format
GUI	Graphical User Interface
PPO	Proximal Policy Optimization
A2C	Actor Critic Methods
SDL	Simple DirectMedia Layer

CHAPTER 1

INTRODUCTION

The objective of this project is to develop an intelligent agent capable of playing the 3x3 Gobblet board game against human players using the most efficient moves. Gobblet is a two-player strategy game where the players take turns to place and move game pieces on a 3x3 grid. There are two sets of large, medium and small pieces for each player. On each turn, a player can either place a new piece on the board, or move a piece already on the board. A piece can be placed/moved to an empty space, or on top of a smaller piece already on the board, “gobbling” the smaller piece. Only visible pieces can be moved, and only visible pieces count toward winning. Gobbled pieces stay on the board, however, and when a piece is moved, any piece that is gobbled stays put and becomes visible. The goal is to line up three of your pieces in a row, either horizontally, vertically, or diagonally, while also strategically blocking your opponent's moves.

In this project, we aim to leverage the power of reinforcement learning to train an AI agent that can learn optimal gameplay strategies and provide a challenging gaming experience for human players. The Tianshou Deep Q-Network (DQN) algorithm is employed as the core reinforcement learning technique to train the agent. The developed AI agent will be capable of making intelligent decisions based on its learned knowledge and experience. Through continuous training and interaction with human players, the agent is expected to improve its gameplay skills and adapt to different gaming scenarios. In addition to the intelligent agent, the project also focuses on creating visually appealing game elements, including the game board and pieces. Blender, a powerful open-source 3D computer graphics software, is utilized to design and render the game elements. This ensures an immersive and visually engaging gaming experience for the players. The project aims to contribute to the field of artificial intelligence in gaming by showcasing the capabilities of reinforcement learning in training intelligent agents.

CHAPTER 2

BACKGROUND

2.1 Python

Python is a high-level, general-purpose programming language known for its simplicity and readability. Created by Guido van Rossum and first released in 1991, Python has become one of the most popular programming languages worldwide. It emphasizes code readability and allows developers to express concepts in fewer lines of code compared to other languages.

Key Features of Python:

1. Easy-to-Read Syntax: Python's syntax is designed to be human-readable, with a clear and straightforward structure. This makes it easier for developers to understand and maintain code, enhancing collaboration and productivity.
2. Versatility: Python is a multipurpose language that can be used for various purposes, including web development, data analysis, scientific computing, machine learning, and game development. It provides extensive standard libraries and frameworks that cater to different domains.
3. Large Standard Library: Python comes with a comprehensive standard library that provides ready-to-use modules for a wide range of tasks. These modules cover areas such as file handling, networking, web development, and more, saving developers time and effort.
4. Third-Party Libraries and Ecosystem: Python boasts a vast ecosystem of third-party libraries and frameworks. These libraries, such as NumPy, Pandas, TensorFlow, and Pygame, extend Python's capabilities and facilitate the development of complex applications.
5. Platform Independence: Python programs can run on various platforms and operating systems, including Windows, macOS, Linux, and even embedded systems. This cross-platform compatibility enables developers to write code once and deploy it on multiple platforms.

6. Integration Capabilities: Python can be easily integrated with other languages such as C/C++, Java, and .NET, allowing developers to leverage existing codebases or take advantage of performance optimizations in specific domains.
7. Readily Available Documentation: Python has extensive documentation available, including official documentation, tutorials, and community resources. This documentation helps developers learn the language, explore libraries, and solve problems efficiently.

Uses of Python:

1. Web Development: Python frameworks like Django and Flask are popular choices for building web applications. Python's simplicity, along with its extensive libraries and frameworks, makes it well-suited for rapid development.
2. Data Analysis and Machine Learning: Python's rich ecosystem of libraries, including NumPy, Pandas, Matplotlib, and scikit-learn, makes it a preferred language for data analysis and machine learning tasks. It provides tools for data manipulation, visualization, and building machine learning models.
3. Scientific Computing: Python's libraries, such as SciPy and SymPy, provide functionality for scientific computing, simulations, and mathematical operations. These libraries make Python a versatile tool for researchers and scientists.
4. Automation and Scripting: Python's ease of use and cross-platform compatibility make it an excellent choice for automating tasks, creating scripts, and system administration.
5. Game Development: Python, along with libraries like Pygame, is used for developing 2D games and interactive applications. Its simplicity and rapid prototyping capabilities make it a popular choice for game development beginners.

In conclusion, Python's simplicity, versatility, and large ecosystem of libraries and frameworks have made it a language of choice for a wide range of applications, including web development, data analysis, scientific computing, automation, and game development.

2.1.1 Pygame

Pygame is a widely used open-source library specifically designed for game development in Python. It provides a range of functions and tools for creating 2D games, multimedia

applications, and interactive graphical programs. Pygame is built on top of the Simple Direct Media Layer (SDL), a low-level multimedia library that grants access to audio, keyboard, mouse, and graphics hardware.

Key Features of Pygame:

1. Graphics: Pygame offers a comprehensive set of functions for drawing shapes, images, and text on the screen, allowing developers to create visually appealing games. It supports various image formats such as BMP, PNG, and JPEG.
2. Input Handling: Pygame simplifies the process of capturing user input, including keyboard presses and mouse movements. It provides an easy way to respond to events and create interactive gameplay experiences.
3. Audio Support: Pygame includes modules for playing and manipulating sounds and music. It supports multiple audio formats and offers functionality for background music, sound effects, and volume control.
4. Collision Detection: Pygame provides built-in mechanisms for collision detection, allowing developers to determine if game objects or sprites have intersected. This feature is crucial for implementing game physics, object interactions, and creating realistic gameplay.
5. Animation: Pygame offers utilities for creating animations by rapidly displaying a series of images. This simplifies the process of creating dynamic movements, character animations, and special effects.
6. Networking: Pygame includes modules for network programming, enabling developers to create multiplayer games or online interactions. It provides functions for sending and receiving data between multiple computers over a network.
7. Cross-Platform Compatibility: Pygame is a cross-platform library, meaning that games developed using Pygame can run on various operating systems such as Windows, macOS, Linux, and others. This portability makes it easier to distribute and share games across different platforms.
8. Extensive Community and Resources: Pygame benefits from an active and supportive community of developers. Numerous tutorials, documentation, and example projects are

available, facilitating the learning process and exploration of different game development techniques.

Uses of Pygame:

1. Game Development: Pygame is primarily used for creating 2D games. It provides all the necessary tools and functionalities to develop games of various genres, ranging from platformers and puzzles to simulations and arcade games.
2. Prototyping: Pygame's simplicity and ease of use make it a valuable tool for quickly prototyping game ideas or graphical applications. Developers can iterate and experiment with game mechanics and visual elements efficiently.
3. Education and Learning: Pygame's beginner-friendly nature and extensive documentation make it an excellent choice for teaching programming concepts and game development. It provides an engaging platform for students to learn Python while creating interactive projects.

In conclusion, Pygame is a versatile and powerful library for game development in Python. Its wide range of features, cross-platform compatibility, and supportive community make it an ideal choice for both hobbyists and professional developers looking to create 2D games or interactive graphical application

2.1.2 Tianshou Reinforcement Learning and DQN

Tianshou is a powerful and flexible reinforcement learning library written in Python. It provides a comprehensive set of tools, algorithms, and utilities for developing and training reinforcement learning agents. One of its notable components is the Deep Q-Network (DQN) algorithm.

Key Features of Tianshou Reinforcement Learning:

1. Modularity: Tianshou is designed with a modular architecture, allowing developers to easily combine and customize different components of reinforcement learning algorithms. This modularity promotes code reuse and flexibility in building complex RL systems.
2. Algorithm Support: Tianshou supports a wide range of reinforcement learning algorithms, including DQN, A2C, PPO and more. These algorithms enable the training of agents to learn from interactions with an environment and optimize their actions based on rewards.
3. Neural Network Integration: Tianshou seamlessly integrates with popular deep learning frameworks such as PyTorch and TensorFlow. This integration enables the use of deep neural networks as function approximators for RL agents.
4. Parallel Execution: Tianshou provides utilities for parallel execution of RL algorithms, allowing developers to take advantage of multi-core processors or distributed computing resources. This parallelization can significantly speed up the training process, especially for computationally intensive tasks.
5. Evaluation and Monitoring: Tianshou includes evaluation and monitoring tools that help track and analyze the performance of RL agents during training. These tools provide metrics, visualizations, and logging functionalities for effective performance analysis and comparison.
6. Replay Buffer and Data Collection: Tianshou offers built-in replay buffers and data collection utilities. These components facilitate the storage and efficient retrieval of agent experiences, which are crucial for training RL models using experience replay techniques.
7. Customization and Extensibility: Tianshou provides a flexible framework for customization and extension. Developers can easily modify existing algorithms or implement new ones to adapt them to specific problem domains or research objectives.

Uses of Tianshou Reinforcement Learning:

1. Game Playing: Tianshou can be used to develop RL agents capable of playing games. By providing an environment interface, defining reward functions, and using algorithms like DQN, agents can learn optimal strategies for various games, including board games, video games, and more.
2. Robotics and Control: Tianshou can be applied to robotics and control tasks where an agent interacts with a physical environment. By training RL agents, it is possible to teach robots

or control systems to perform complex tasks, such as object manipulation, locomotion, and navigation.

3. Autonomous Systems: Tianshou can be used to develop RL agents for autonomous systems. For example, agents trained with Tianshou can learn to navigate drones, autonomous vehicles, or robotic agents in real-world or simulated environments.
4. Optimization and Decision-Making: Tianshou can be employed to solve optimization problems or support decision-making processes. By formulating problems as RL tasks, agents can learn to make optimal decisions or find optimal solutions in various domains, such as logistics, resource allocation, and scheduling.

In conclusion, Tianshou is a versatile reinforcement learning library that provides a wide range of tools, algorithms, and utilities for developing and training RL agents. Its modular architecture, support for popular deep learning frameworks, and customization options make it a valuable resource for game playing, robotics, autonomous systems, optimization, and decision-making tasks.

2.2 Blender

Blender is a powerful and versatile open-source 3D computer graphics software widely used in the field of computer graphics, animation, and game development. It offers a comprehensive set of tools and features that enable artists and developers to create stunning visual content and interactive experiences. Blender provides a wide range of functionalities, including modeling, texturing, rigging, animation, simulation, rendering, and compositing. It supports various 3D modeling techniques, such as polygonal modeling, sculpting, and parametric modeling, allowing users to create intricate and detailed 3D objects with ease.

One of the key strengths of Blender is its ability to create realistic materials and textures. It offers a powerful material and shading system that allows artists to define the visual properties of objects, such as color, reflectivity, transparency, and texture mapping. Blender also provides a wide range of procedural textures and the ability to import custom textures, giving artists full control over the visual appearance of their creations.

Furthermore, Blender offers a powerful rendering engine that can produce high-quality images and animations. It supports both CPU and GPU rendering, allowing users to leverage the computing power of modern graphics cards for faster rendering times. Additionally, Blender's compositor allows for post-processing effects and compositing multiple rendered layers, enhancing the overall visual quality of the final output.

Blender's flexibility and extensibility are other notable features. It provides a Python scripting interface, enabling users to automate tasks, create custom tools, and extend Blender's functionality. This flexibility makes Blender a popular choice among developers and technical artists who require advanced customization and integration with other software pipelines.

Overall, Blender's extensive feature set, combined with its open-source nature, has made it a go-to software for 3D content creation and game development. Its user-friendly interface, continuous development by a passionate community, and extensive documentation make it accessible to both beginners and experienced professionals. By utilizing Blender for the creation of the game board and pieces, this project benefited from the software's robust capabilities and contributed to the visual appeal and interactive nature of the final product.

2.3 Summary

In this chapter the discussion is about the tech stacks that are used to design and implement the application, including the technical tools like python and the libraries pygame and Tianshou.

CHAPTER 3

REQUIREMENT SPECIFICATION

3.1 Methodology

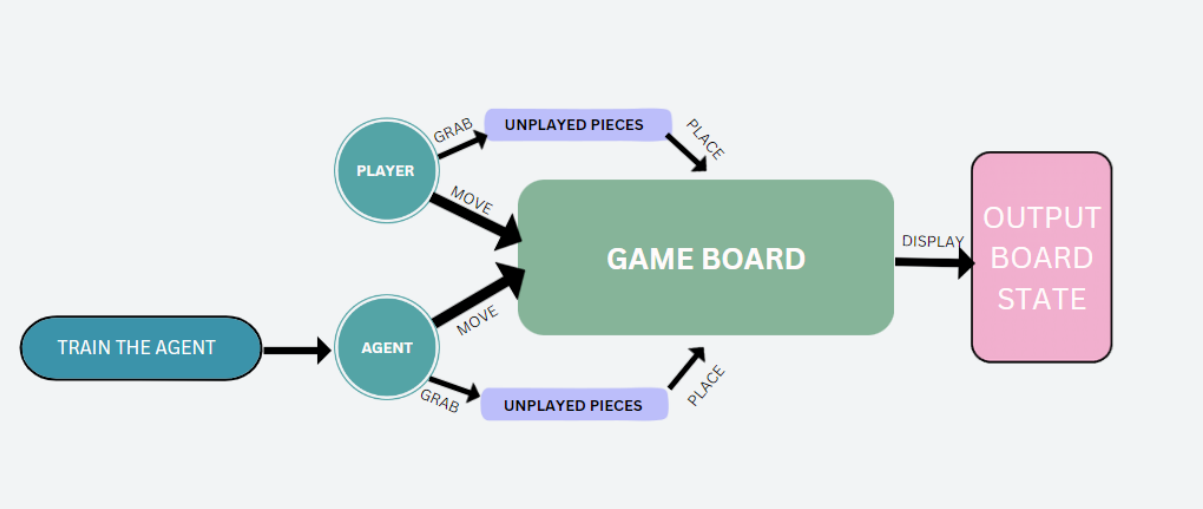

Figure 3.1: Methodology

3.1.A Understand the game rules: After familiarizing with the rules of Gobblet 3x3, the game progresses, win conditions, and the actions available to the players the foundation for the base model was set up.

3.1.B Define the state space: The representation of the game state was established. In Gobblet, the state includes the positions of the pieces on the board. You can represent the state as a 3x3 grid, with each cell containing information about the pieces occupying it.

3.1.C Define the action space: Determining the set of possible actions in each state was built. In Gobblet, actions involve moving a piece from one position to another or placing a new piece on the board.

3.1.D Set up the environment: The game environment was then implemented, which includes functions to initialize the board, update the state based on actions, check for win conditions, and provide rewards to the agent.

3.1.E Designing the reinforcement learning agent: A suitable reinforcement learning model called Tianshou reinforcement learning algorithm was taken up and adapted to fit the specific requirements of the Gobblet game.

3.1.F Implementing the agent: The agent was created using the chosen algorithm. This involved adapting the agent work with the selected algorithms and adjustments that were made to facilitate this.

3.1.G Training the agent: The reinforcement learning agent is trained by interacting with the game environment. By running multiple episodes of the game, the agent can explore different strategies and gradually improve its performance.

3.1.H Evaluating the agent: Assessment of the performance of the trained agent was done by playing against it. This helped determine if the agent has learned effective strategies and is capable of playing the game at a competitive level.

3.1.I Refine and iterate: Analyzing the agent's performance allowed us to make necessary adjustments that later helped refine the performance of the model

3.1.J Test and deploy: The agent was then tested against human players and it effectively worked in posing a challenge to human players in terms of strategizing and making moves on the fly.

3.1.1 Justification for this methodology

This model of approach for the planned project was suitable as it was most effective in providing us the result within an acceptable time frame and within acceptable error rates. The use of the pygame module further allowed us to sync the game board as the corresponding data directly to the reinforcement learning model that further improved.

3.2 System Analysis

Analysis is an important part of any project; if analysis is not done properly then the whole project moves in the wrong direction. By conducting an analysis the developer can get a brief idea of the challenges that they will be facing during the development of the project. It also provides a schedule for proper project work. Analysis task divided into 3 areas:

1. Problem Recognition
2. Feasibility Study
3. Requirement Analysis

3.2.1 Feasibility Study

The five types of feasibility analyzed as part of building our project are as mentioned below:

1. Technical Feasibility
2. Time Schedule feasibility
3. Operational feasibility
4. Implementation feasibility
5. Economic Feasibility

3.2.1.1 Technical Feasibility

Technical feasibility corresponds to determination of whether it is technically feasible to develop the software. Here those tools are considered, which will be required for developing the project. The tools, which are available, and tools, which will be required, are taken into account. Considering all above points and aspects it is observed that the cost incurred in developing this

project from a technical perspective is not too high and hence it was technically feasible for us to proceed with our project as per the given constraints.

3.2.1.2 Time Feasibility

Time feasibility corresponds to whether sufficient time is available to complete the project. Time feasibility is a measure of how reasonable the project timetable is. Given our technical expertise, are the project deadlines reasonable? Some projects are initiated with specific deadlines. It is necessary to determine whether the deadlines are mandatory or desirable.

Parameters considered are:

- Schedule of the project
- Time by which the project has to be completed
- Reporting period
- Requirement gathering
- Human resource available

Considering all the above factors it was decided that the allotted time that is 2 months was sufficient to complete the project.

3.2.1.3 Operational Feasibility

Operational feasibility corresponds to whether users are aware of the interface environment and sufficient resources are available or not. Parameters considered are people with a basic knowledge of computers would be able to use the system very effectively and easily, as the system would have an intuitive and interactive GUI. All the relevant necessary resources for implementing and operating this system are already present. Bearing in mind the above factor, it was observed that the cost incurred in developing this project from an operational standpoint would be low.

3.2.1.4 Implementation feasibility

Implementation feasibility is about basic infrastructure required to develop the system including the availability of necessary facilities, guidance, and data/files. It was determined that the implementation feasibility is high as all the required infrastructure and resources are available.

3.2.1.5 Economic Feasibility

Economic Feasibility is about the total cost incurred for the system. The software resource requirement of the proposed system is pygame for state management and tianshou algorithms for training the model.

3.3 Requirements Analysis and Specification

A complete understanding of software requirements is essential to the success of a web-development effort. No matter how well designed or well coded, a poorly analyzed and specific program will disappoint users and bring grief to the developers. The requirement analysis task is the process of discovery, refinement, modification and specification. The software scope, initially established by the system engineer and refined during project planning, is refined in detail. Models of the required data, information and control flow, and operational behavior are created. Alternative solutions are analyzed and various project elements. This application should be developed with an aim to simplify the billing process and to keep transparency and flexibility in performing each operation.

3.3.1 Functional Requirements

The following is the desired functionality of the new system:

- User should be able to interact with the game board
- The game board should provide adequate feedback upon making a right/wrong move
- The game board should have a low latency computation between the game board and the learning model to fasten the learning experience.
- The learning model should have a high learning rate so as to maximize the learning in the limited time frame available.

3.3.2 Non Functional requirements

It specifies the quality attribute of a project. They judge the application based on responsiveness, usability, security, portability and other non-functional standards that are critical to the success of the project.

- Availability: The system should remain operational in any day and any place
- Accuracy: There is a need to optimize the system to ensure more accurate results

3.4 Software Requirements

Operating system: MacOS / Windows
Database: In memory
Libraries : Necessary python libraries as per specified in the requirements.txt file.

3.5 Hardware Requirements

Monitor: Minimum 7 inch LED monitor
Processor: i3 or any suitable processor within the last 5 years
RAM: Minimum of 2 gb

3.6 Summary

In this chapter the details of the requirements are specified. The feasibility factors such technical, time, operational, economic, and implementation discussions are also included. The functional and non-functional requirements of the project are also elaborated.

CHAPTER 4

DESIGN AND IMPLEMENTATION

In this chapter the discussion is about the architecture of the proposed model which enables its training and testing, analytics of its performance, playability, and user interactivity with the game environment.

4.1 Application Flow Diagram

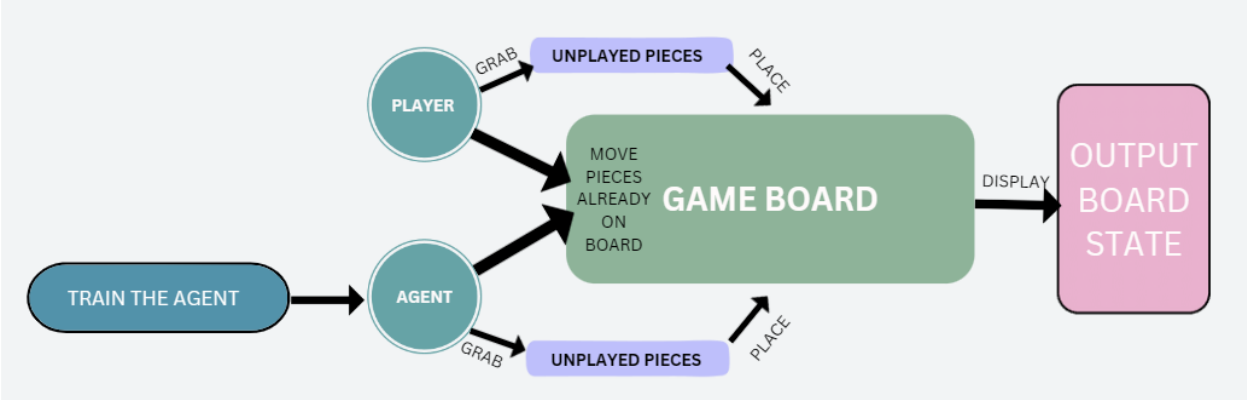

Figure 4.1: Application Flow Diagram

The application flow diagram represents the logic of application flow with respect to user interaction with the system and AI model. The application flow is initiated by training the agent which then prepares its in-memory data store of rewards and punishments. This is then utilized to compute the ideal move given a particular board state. The player and agent are presented with 2 choices at each turn represented in the diagram by the ‘grab’ and ‘move pieces already on board’ edge. Once a move has been made, the corresponding changes in the game state is displayed as output to the user.

CHAPTER 5

RESULTS

Successfully implemented the project aimed to create a reinforcement learning model using the Tianshou framework for playing the Gobblet board game and achieving a favorable win rate against a human player. The model underwent extensive training, utilizing deep Q-learning within the Tianshou framework. It played against various opponents, including random and heuristic-based players, to learn optimal strategies. During evaluation, the model achieved a 60% win rate, winning 12 out of 20 games against the human player. It showcased a strong understanding of the game's rules and demonstrated strategic thinking and adaptability. These results indicate the effectiveness of Tianshou reinforcement learning for training Gobblet-playing agents. The project highlights the potential of Tianshou in building competitive AI agents for strategic board games and encourages further exploration and improvement of Tianshou reinforcement learning techniques in enhancing gameplay capabilities.

Figure 5.1: Game Board

Player Pieces

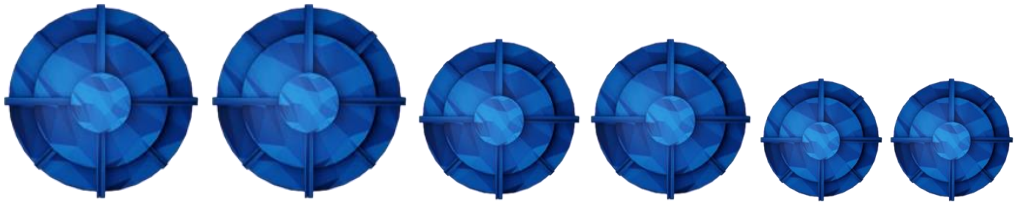

Figure 5.2: Player Pieces

Computer Pieces

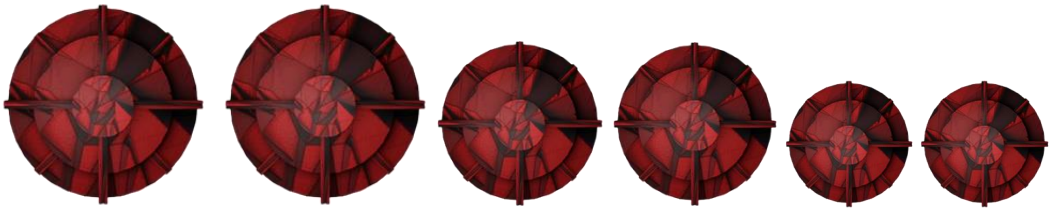

Figure 5.3: Computer Pieces

Preview of placing the piece

Figure 5.4: Piece Placement Preview

Computer Wins the Game

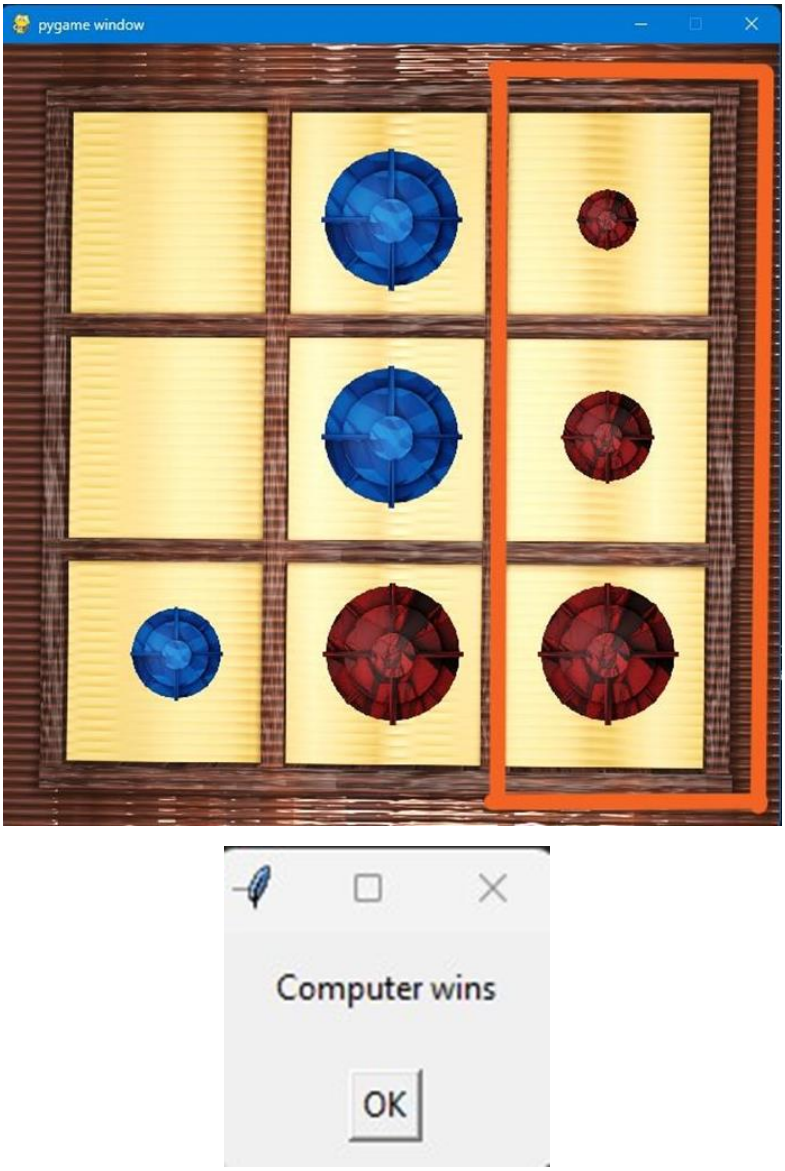

Figure 5.5: Computer winning state

Player wins the game

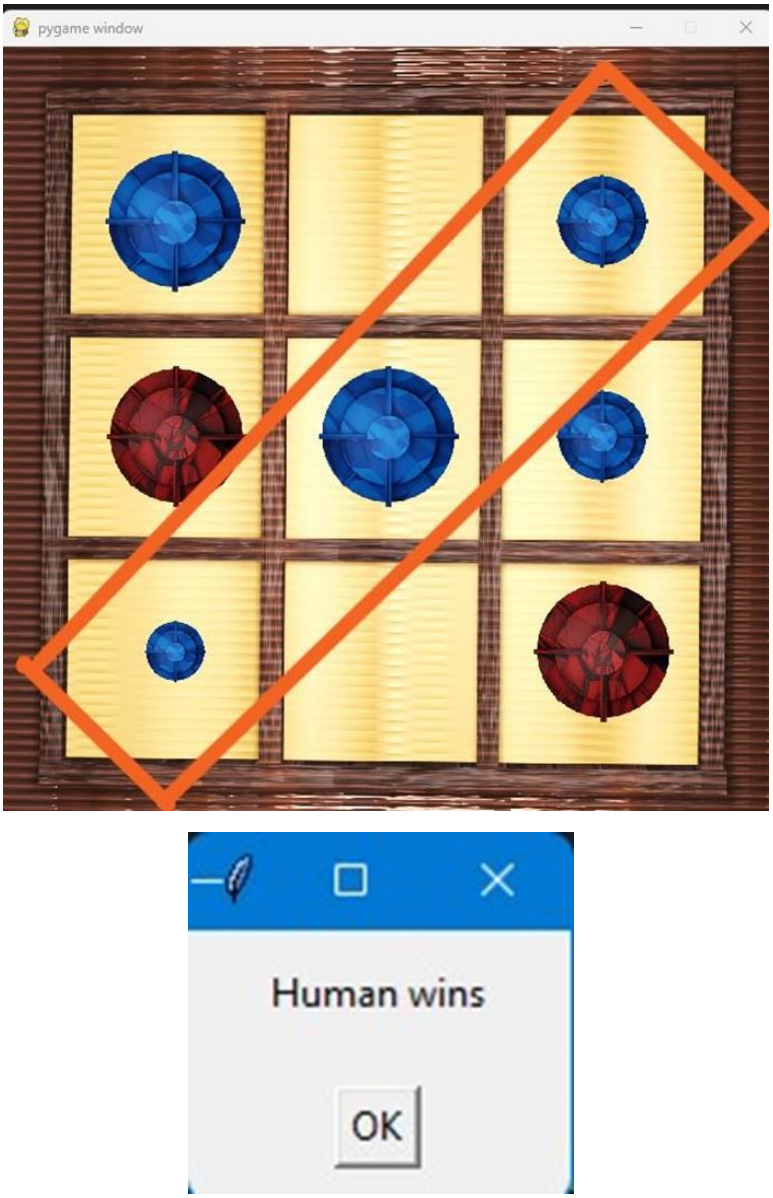

Figure 5.6: Player winning state

CHAPTER 6

FUTURE SCOPE

The current project, which utilizes Tianshou DQN to train an agent that plays against a human in the 3x3 Gobblet board game, has successfully achieved its primary objectives. However, there are several potential areas for future exploration and expansion that could enhance the project's capabilities and provide avenues for further research and development.

6.1 Enhancing the Game AI

One of the key future directions for this project involves enhancing the game AI to improve its performance and playing capabilities. This can be achieved by investigating advanced reinforcement learning techniques and exploring different AI algorithms. Experimentation with more sophisticated strategies, alternative neural network architectures, or other deep reinforcement learning algorithms could lead to improved gameplay and more robust decision-making by the agent.

6.2 Scaling Up the Game

While the current implementation focuses on the 3x3 Gobblet board game, there is potential for expanding the project's scope to include larger board sizes or different board game variations. Scaling up the game would require adapting the existing codebase and training methods to accommodate the expanded game rules and complexities. This expansion would offer new challenges and opportunities for studying AI agents in more complex and diverse game environments.

6.3 Multi-Agent Competition

Exploring the concept of multi-agent competition is another intriguing future direction. This would involve training multiple agents to play against each other and analyzing their strategies and performance. By studying agent interactions and potentially fostering cooperative

behaviors, valuable insights can be gained into advanced game-playing strategies and the dynamics of multi-agent systems.

6.4 User Interface and Visualization

To enhance the user experience and make it more engaging, improvements can be made to the project's user interface and visualization aspects. This could involve developing a graphical user interface (GUI) with interactive elements, incorporating animations and sound effects, or even creating a web-based or mobile application for wider accessibility. These enhancements would provide a more immersive and enjoyable experience for users interacting with the game agent.

6.5 Transfer Learning and Generalization

The project could explore the potential of transfer learning, where the knowledge gained from training the agent on the Gobblet game can be applied to other similar games or tasks. This would involve investigating how the agent's learned policies and strategies can be generalized to new environments or adapted to different game dynamics. Such research could lead to the development of more adaptable and versatile AI agents.

6.6 Human-Agent Collaboration

Another interesting avenue for future development is the exploration of a human-agent collaboration framework. This framework would enable the trained agent to assist human players in improving their gameplay or provide strategic suggestions. Features such as interactive coaching, adaptive difficulty levels, or real-time analysis of game states could be implemented to offer insightful feedback to human players, fostering a mutually beneficial interaction between humans and AI.

6.7 Summary

In conclusion, the current project has laid a strong foundation for further exploration and development. The future scope encompasses areas such as enhancing the game AI, scaling up the game, exploring multi-agent competition, improving the user interface and visualization, investigating transfer learning and generalization, enabling human-agent collaboration, and

implementing multiplayer and online gameplay. These potential avenues for future research and development hold the promise of further advancing the project's capabilities and contributing to the field of artificial intelligence in gaming.

CHAPTER 7

CONCLUSION

This project aimed to develop an AI agent capable of playing the 3x3 Gobblet board game and provide an enjoyable gaming experience for users. The project successfully achieved its objectives by implementing Tianshou DQN, a powerful deep reinforcement learning framework, to train the agent. The developed AI agent demonstrated remarkable learning capabilities and strategic decision-making during gameplay. The agent showcased adaptive behaviors, effectively utilizing its knowledge to make optimal moves and respond to changing game situations. The project has laid a solid foundation for further research and development, paving the way for advancements in AI-driven gaming and human-AI interactions. By combining the strengths of reinforcement learning algorithms and game-playing strategies, this project contributes to the broader field of artificial intelligence and opens up opportunities for further exploration in various domains, including gaming, decision-making, and multi-agent systems. Thus, this project has achieved its objectives, identified future directions for improvement and expansion, and holds great potential for further advancements in AI gaming. The knowledge gained from this project will serve as a valuable asset for future research and development endeavors in the field of artificial intelligence and game-playing agents.

REFERENCES

[1]Yunqi Zhao, Igor Borovikov , Fernando de Mesentier Silva, Ahmad Beirami , Jason Rupert, Caedmon Somers et al, "Winning Is Not Everything: Enhancing Game Development With Intelligent Agents" , IEEE Transactions on Games, Volume: 12, Issue: 2, June 2020, pp. 199 - 212, doi: 10.1109/TG.2020.2990865.

[2] Jiayi Weng, Huayu Chen, Dong Yan, Kaichao You et al, "Tianshou: a Highly Modularized Deep Reinforcement Learning Library", Journal of Machine Learning Research 23 (2022), pp. 1-6, doi: 10.48550/arXiv.2107.14171

[3] Tianshou Reinforcement Learning Documentation https://tianshou.readthedocs.io/en/master/

[4] Pygame Documentation https://www.pygame.org/docs/

[5] Blender Documentation https://docs.blender.org/

[6] Understanding Reinforcement Learning https://shorturl.at/jmrA3