Project Report

Project Title: Game Playing Agent
Course: Artificial Intelligence (Mc470302)
Semester: 3rd
Branch: Mca (Ai And Iot)
Department: Computer Science And Engineering
Institution: National Institute Of Technology (Nit), Patna
Team Members:

- Prashant Kumar Mishra (2447021)
- Rishi Kumar (2447031)
- Satyam Bhardwaj (2447051)

1. Introduction

In today's World Artificial Intelligence (AI) has made a remarkable progress in the domain of game playing. From early rule-based systems such as IBM's Deep Blue, On May 11, 1997, defeated the reigning world chess champion, Garry Kasparov, in a six-game match, to modern reinforcement learning (RL) agents like alphago and alphazero, AI-based game playing has become a powerful testbed for developing general-purpose intelligent systems.

In this project, we aim to design and implement a **multi-agent game playing system** for the game of **Pacman**, using **Deep Learning** and **Reinforcement Learning (RL)**. Unlike traditional rule-based implementations, our system will use **image-based game board states** and past training experiences to learn strategies autonomously.

The project will involve three key components:

- **Frontend (GUI Simulation):** For displaying the game board and interactions between agents.
- **Backend (Game Engine):** For handling rules, moves, and multi-agent interactions.
- **Core Learning Model (AI Agent):** A reinforcement learning agent (Deep Q-Networks and possible extensions like Double DQN, Dueling DQN, or CNN + RNN) trained to maximize performance in Pacman by learning from visual states.

This project not only provides practical implementation experience in AI but also strengthens our understanding of reinforcement learning, deep learning, and multi-agent systems.

2. Objectives

The primary objectives of the project are:

- **To implement a multi-agent Pacman game environment** that supports interactions between the Pacman agent and adversarial ghost agents.
- **To develop a reinforcement learning-based Pacman agent** that can learn from image-based game states rather than handcrafted features.
- **To experiment with deep learning models (CNN, DQN, Double DQN, and Dueling DQN)** for state representation and decision-making.
- **To design a user-friendly GUI frontend** for simulating the game, allowing real-time observation of agent training and performance.
- **To integrate backend game logic** for smooth execution and training of multiple agents in the environment.
- **To document the system architecture, implementation details, and experimental results** .

3. Hardware and Software Requirements

- **3.1) Hardware Requirements**
  - **Processor:** Intel i5/i7 (or AMD equivalent), minimum 2.4 ghz
  - **RAM:** Minimum 8 GB (16 GB recommended for deep RL training)
  - **GPU:** NVIDIA GPU with CUDA support (e.g., GTX 1050 Ti or higher) – recommended for faster training
  - **Storage:** Minimum 20 GB free space for datasets, models, and logs
- **3.2) Software Requirements**
  - **Programming Language:** Python 3.8 or above
  - **Libraries & Frameworks:**
    - Pytorch / tensorflow (for deep learning models)
    - Openai Gym and Gym[Atari] (for environment simulation)
    - Python-Chess / Pygame (for GUI and environment handling)
    - Numpy, Pandas, Matplotlib (for data handling and visualization)
  - **Development Tools:**

        - Jupyter Notebook / VS Code / pycharm
        - Github (for version control and collaboration)
    - **Operating System:**
        - Windows 10/11 or Ubuntu 20.04 LTS or macos 15.5

---

4. Methodology

This project will follow a structured workflow:

4.1 Game Environment Design

- Use of **Pygame** and **openai Gym** to simulate a Pacman environment.
- Defining **multi-agent interaction rules**:
    - Pacman (player agent) aims to maximize score by eating pellets, capsules, and avoiding ghosts.
    - Ghosts (adversarial agents) aim to minimize Pacman's survival time.

4.2 State Representation

- Conversion of the game board into **image-based states** (grid representation or pixel representation).
- Use of **Convolutional Neural Networks (cnns)** to extract spatial features from game states.

4.3 Reinforcement Learning Approach

- Implementation of **Deep Q-Learning (DQN):**
    - Using experience replay to store past transitions (state, action, reward, next state).
    - Application of ε-greedy exploration with decaying ε for balancing exploration vs exploitation.
- Extending to **Double DQN** to reduce overestimation bias.
- Experimentation with **Dueling DQN architecture** to separately estimate state-value and advantage functions.

4.4 Training Process

- Training the Pacman agent against ghost agent using different training approach/strategies.
- Evaluation of performance using different metrics such as:
    - Win rate

    - Average reward per episode
    - Learning curve (reward vs steps)

4.5 Frontend and Visualization

- Developing a **GUI in Pygame** for real-time visualization of the training process.
- Implementation of log visualizations (reward graphs, agent progress) using **Matplotlib**.

4.6 Documentation

- Recording all design decisions, algorithms used, experiments conducted, and results obtained.
- Preparation of a detailed final project report and presentation slides.

---

5. Responsibility Distribution

Prashant Kumar Mishra (2447021)

- Role: **Model Architect & Documentation**
- Responsibilities:
    - Designing the reinforcement learning models (CNN, DQN, Double DQN, Dueling DQN).
    - Implementation of model training and optimization.
    - Maintaining documentation of methods, algorithms, and results.

Rishi Kumar (2447031)

- Role: **Backend & Environment**
- Responsibilities:
    - Implementation of Pacman game environment and ghost agent logic using Pygame/Gym.
    - Handling the state representation (grid → image conversion).
    - Integration of backend with model training loop.

Satyam Bhardwaj (2447051)

- Role: **Frontend & Visualization**
- Responsibilities:
    - Developing a GUI frontend for simulation of Pacman training.

    - Visualization of agent performance using graphs, logs, and metrics.
    - Assist in devloping testing models and debugging integration issues.

---

6. Conclusion

- This project aims to combine deep learning, reinforcement learning, and multi-agent system design to create a fully functional **game playing agent** for Pacman.
- The use of **image-based state input** ensures that the system does not rely on handcrafted features, making the problem closer to real-world AI challenges.
- With a modular approach, clear task division, and integration of GUI + backend + RL models, this project will not only strengthen our understanding of AI concepts but also showcase a practical implementation of **game-playing agents in action**.