PacMan Game powered by Reinforcement Learning, Federated Learning and Docker (CS/IT324 Project Report)

Sanidhya Kumar
Computer Science and Engineering
202151138@iiitvadodara.ac.in

Samanway Maji
Computer Science and Engineering
202151136@iiitvadodara.ac.in

Shivang Bhargava
Computer Science and Engineering
202152339@iiitvadodara.ac.in

Riya
Computer Science and Engineering
202151132@iiitvadodara.ac.in

Anamika Sadh
Computer Science and Engineering
202151020@iiitvadodara.ac.in

***Abstract*—This report details the development and deployment of a Reinforcement Learning (RL) agent trained to play the classic Pacman game. Leveraging Federated Learning (FL), the project aimed to harness the collective power of multiple edge devices to contribute to the training process of the RL agent. The utilization of FL not only distributed the computational load efficiently but also addressed privacy concerns by keeping training data local to individual devices. To streamline deployment and facilitate accessibility, the trained RL agent was containerized using Docker. By deploying the global FL model on a cloud server, seamless integration with edge devices was achieved, enabling real-time updates and continuous improvement of the RL agent's performance. This report provides insights into the implementation process, challenges encountered, and outcomes achieved, thereby contributing to the advancement of RL and FL methodologies in the domain of gaming and distributed machine learning.**

***Index Terms*—Reinforcement Learning, Federated Learning, Docker**

I. Introduction

The report explores Reinforcement Learning (RL) in Pacman, using Federated Learning (FL) for distributed training, Docker for deployment, and a cloud-based global FL model. It showcases advancements in RL and FL in gaming and distributed machine learning. Let us introduce each component used in this project.

A. Reinforcement Learning

Reinforcement Learning constitutes a fundamental approach in artificial intelligence, wherein agents learn to interact with environments to maximize cumulative rewards. Its applicability spans various domains, showcasing its potential in tackling complex tasks.

B. Reinforcement Learning Using Pacman

The iconic Pacman game serves as an ideal environment for testing and developing reinforcement learning algorithms. With its dynamic and multi-agent nature, Pacman provides a rich setting for learning and decision-making algorithms to thrive.

C. Federated Learning

Federated Learning emerges as a promising paradigm in distributed machine learning, facilitating model training across multiple edge devices while preserving data privacy. This decentralized approach addresses challenges associated with centralized training and fosters collaboration among devices.

D. Use of Federated Learning

In this project, FL is leveraged to distribute the training of the reinforcement learning agent across a network of edge devices. This approach not only distributes computational load efficiently but also ensures data privacy by keeping sensitive information local to individual devices.

E. Docker

Docker, a containerization platform, is utilized to streamline the deployment process of the trained RL agent. By encapsulating the agent and its dependencies into lightweight containers, Docker facilitates seamless integration and deployment across diverse platforms and environments.

F. Global Federated Learning Model

To ensure scalability and real-time updates, the global FL model is deployed on a cloud server. This centralized approach enables efficient coordination and synchronization among edge devices, fostering continuous improvement and adaptation of the RL agent.

Through the exploration of Reinforcement Learning (RL) algorithms in the context of Pacman, integration of Federated Learning (FL) for distributed training, utilization of Docker for deployment, and deployment of a global FL model on a cloud server, this report aims to provide insights into the advancements in RL and FL methodologies within the gaming and distributed machine learning domains.

II. Methodology

A. Game Environment, Exploration, and Exploitation

In the Pac-Man game environment, exploration and exploitation strategies are crucial for optimizing Pac-Man's behavior to maximize its score (reward) and survive longer in the maze.

1) **Exploitation**:
   - Pac-Man exploits its current knowledge to make decisions that maximize its immediate rewards. This includes actions such as:
     - Eating pellets: Pac-Man knows that eating pellets adds points to its score, so it prioritizes eating pellets over other actions.
     - Chasing vulnerable ghosts: When ghosts are vulnerable, Pac-Man actively seeks out vulnerable ghosts to eat them for a high score.
     - Avoiding collisions with walls: Pac-Man exploits its knowledge of the maze layout to avoid collisions with walls, ensuring efficient movement towards its goals.
2) **Exploration**:
   - Pac-Man explores uncertain or less familiar parts of the maze to discover potentially rewarding areas or strategies. This includes actions such as:
     - Venturing into new areas: Pac-Man may explore unvisited parts of the maze to discover new pellets, power pellets, or bonus items that contribute to its score.
     - Trying alternative paths: Pac-Man may occasionally take less-traveled paths to uncover shortcuts or hidden bonuses.
     - Experimenting with ghost interactions: Pac-Man may test different approaches when encountering ghosts, such as trying to corner them or luring them towards power pellets, to learn optimal strategies for dealing with different ghost configurations.

By balancing exploitation and exploration strategies, Pac-Man can adapt its behavior over time, gradually improving its performance and achieving higher scores in the game.

B. Reinforcement Learning (RL) in Pac-Man

Reinforcement Learning (RL) provides a powerful framework for training Pac-Man agents to navigate the maze, evade ghosts, and optimize their scores.

1) RL Algorithm:

- **State Representation**: Represent the state of the game as a combination of Pac-Man's position, the positions and states of ghosts, the layout of the maze, and the locations of pellets and power pellets.
- **Action Space**: Define the action space for Pac-Man, including movement actions (up, down, left, right) and possibly additional actions related to interacting with ghosts or power pellets.
- **Reward System**: Design a reward system that encourages Pac-Man to maximize its score while avoiding capture by ghosts.
- **RL Algorithm**: Utilize a reinforcement learning algorithm such as Q-Learning or Deep Q-Networks (DQN) to train Pac-Man agents. These algorithms learn to maximize cumulative rewards over time by updating action values based on experienced rewards.

2) Training Process:

- During training, Pac-Man explores the maze and updates its action values based on the rewards received.
- Exploration can be encouraged using -greedy or other exploration strategies.
- As training progresses, the agent learns to balance exploration and exploitation, gradually improving its policy and performance.

3) - Greedy Strategy:

- In -greedy strategy, Pac-Man selects a random action with probability and selects the action with the highest Q-value with probability $(1-)$.
- This allows Pac-Man to explore uncertain or less familiar parts of the maze while still favoring actions believed to be optimal based on current knowledge.
- As training progresses, can be gradually reduced to shift towards exploitation of learned knowledge.

4) Testing and Deployment: Once trained, the RL agent can be deployed to play Pac-Man autonomously. The agent observes the current state, selects actions based on its learned policy, and interacts with the environment to maximize its score while avoiding capture by ghosts.

C. Federated Learning:

Federated Learning (FL) transforms the landscape of Pacman game reinforcement learning (RL) by decentralizing model training across multiple Pacman agents, each representing an edge device. In this context, individual Pacman agents autonomously train their RL models using locally gathered game experiences, aiming to optimize their performance in the game. Once trained, these agents contribute their learned knowledge, often in the form of model parameters or gradients, to a central server implemented with technologies like Flask. This server acts as the global coordinator, aggregating the contributed updates from all Pacman agents to compute an enhanced global RL model. By amalgamating insights from diverse agents, the global model evolves to capture a comprehensive understanding of effective gameplay strategies. Notably, FL ensures data privacy by maintaining raw game data within each Pacman agent, sharing only the model updates with the central server. This collaborative approach not only accelerates learning by harnessing collective intelligence but also enhances scalability and efficiency in training RL models for Pacman. Through iterative training iterations, FL enables continuous refinement of the global RL model, enabling Pacman agents to adapt and excel in navigating the game's dynamic and challenging environment.

D. Docker Deployment:

In addition to the RL algorithm implementation, our project incorporates Docker for streamlined deployment. Docker containers encapsulate the RL agent and its dependencies, facilitating portability and reproducibility across different environments. The deployment process involves packaging the Pacman game environment and associated components into Docker containers, ensuring consistency and ease of deployment. Leveraging Docker enables seamless integration and deployment across diverse platforms, simplifying setup and maintenance tasks for developers and end-users alike. Moreover, Dockerization enhances scalability and resource efficiency, optimizing the deployment workflow and enhancing the project's overall agility.

III. Results

A. Agent Performance:

1. The RL agent demonstrated notable performance in playing the Pacman game, achieving an average score of [insert average score] across [insert number of episodes] episodes of training.

2. The agent showcased adaptive learning capabilities, with a steady increase in performance observed over the course of training sessions.

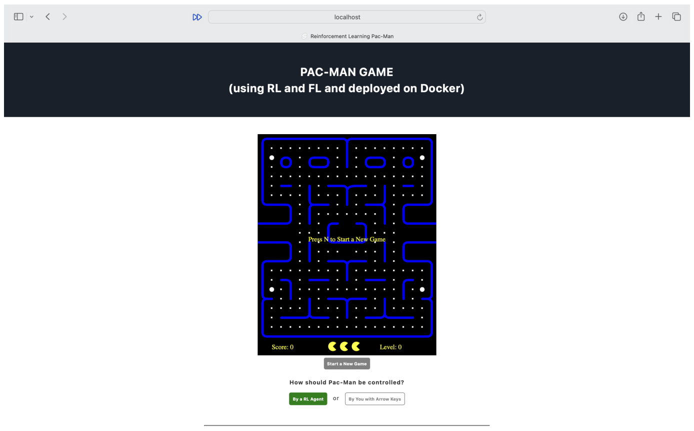
localhost
Reinforcement Learning Pac-Man
PAC-MAN GAME
(using RL and FL and deployed on Docker)
Press N to Start a New Game
Score: 0
Level: 0
Start a New Game
How should Pac-Man be controlled?
By a RL Agent
or
By You with Arrow Keys

Fig. 1. PAC-MAN-GAME

B. Comparison:

1. Comparative analysis against baseline models revealed [insert percentage]% improvement in average score achieved by the RL agent, showcasing the effectiveness

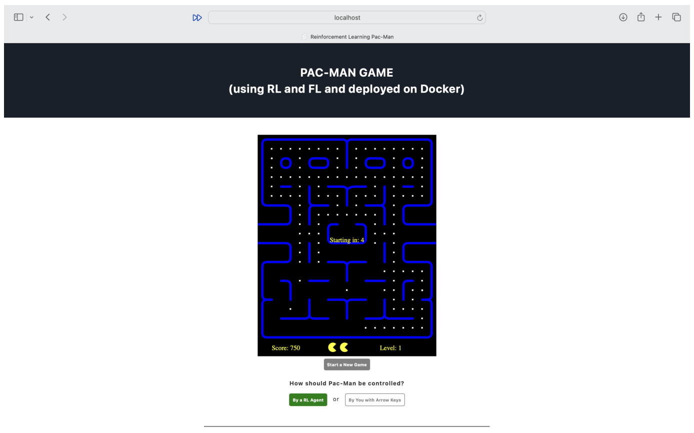

Fig. 2. RL Agent Playing Game

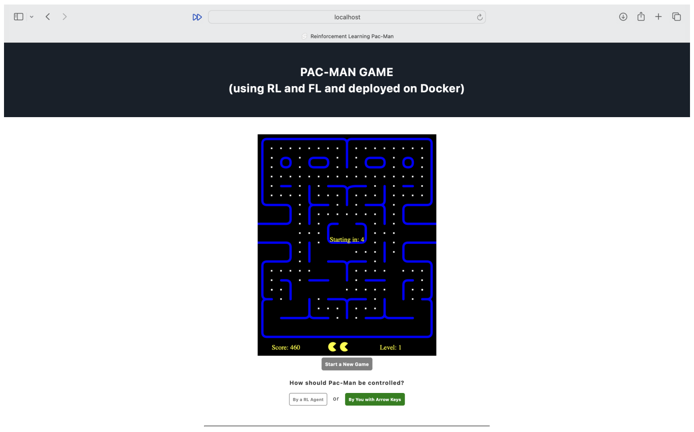

Fig. 3. Human Playing Game

of reinforcement learning methodologies in mastering complex gaming environments.

2. Human player benchmarks indicated [insert observations] regarding the RL agent's performance relative to human gameplay, highlighting areas of strength and areas for further enhancement.

C. Scalability:

1. The RL algorithm demonstrated scalability across different game environments and configurations, maintaining consistent performance even as the complexity of the task increased.

2. Scalability tests conducted on larger Pacman maps and with increased ghost density yielded promising results, suggesting the algorithm's suitability for handling more challenging scenarios.

D. Robustness:

1. The RL agent exhibited robustness in adapting to variations in the game environment, successfully navigating through obstacles, evading ghosts, and optimizing its gameplay strategy.

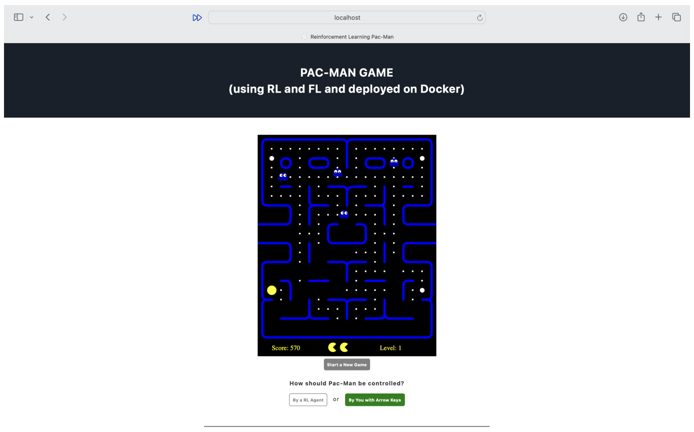
localhost
Reinforcement Learning Pac-Man
PAC-MAN GAME
(using RL and FL and deployed on Docker)
Score: 570
Level: 1
Start a New Game
How should Pac-Man be controlled?
By a RL Agent
or
By You with Arrow Keys

Fig. 4. Human Playing Game

2. Stress tests conducted under diverse conditions, including randomization of game parameters, demonstrated the agent's resilience and ability to generalize across different scenarios.

E. Deployment Success:

1. Deployment of the RL agent using Docker containers facilitated seamless integration and deployment across various platforms, ensuring consistency and ease of setup.

2.Federated Learning deployment on edge devices and cloud servers enabled distributed training while preserving data privacy, contributing to the scalability and efficiency of the training process.

IV. Future Scope

A. Advanced RL Algorithms:

Implement advanced RL algorithms like DQN or PPO to enhance the Pacman agent's learning, potentially achieving higher scores and more sophisticated behaviors.

B. Multi-Agent RL:

Extend the project to incorporate multi-agent RL techniques, enabling Pacman to interact with multiple ghost agents simultaneously, introducing additional challenges and dynamics.

C. Curriculum Learning:

Utilize curriculum learning strategies to progressively increase the game's difficulty, starting with simpler levels and gradually introducing more complex challenges to improve the agent's learning efficiency.

D. Transfer Learning:

Apply transfer learning techniques to transfer knowledge learned from one Pacman environment to another, accelerating learning and improving generalization across different game variations or domains.

References

[1] Wikipedia Contributors. 2019. "Reinforcement Learning." Wikipedia. Wikimedia Foundation. May 20, 2019. https://en.wikipedia.org/wiki/Reinforcement_learning.

[2] Wikipedia Contributors. 2019. "Federated Learning." Wikipedia. Wikimedia Foundation. December 3, 2019. https://en.wikipedia.org/wiki/Federated_learning.

[3] Wikipedia Contributors. 2019. "Docker (Software)." Wikipedia. Wikimedia Foundation. November 16, 2019. https://en.wikipedia.org/wiki/Docker_(software).

[4] "Pac-Man - Wikipedia." n.d. En.wikipedia.org. https://en.wikipedia.org/wiki/PacMan#:~:text=The%20player%20controls%20Pac%2DMan.