भारतीय सूचना प्रौद्योगिकी अभिकल्पना,
एवं विनिर्माण संस्थान, कांचीपुरम

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY,
DESIGN AND MANUFACTURING, KANCHEEPURAM

Reinforcement Learning Project Report

Comparing performance of A3C and DQN on Breakout(Atari) game

Team Members

Sl. No.	Name	Roll Number
1	Vishnu Teja Surla	CS21B2037
2	Sai Kiran Bondi	CS21B2030
3	Veera Abhiram Palakondu	CS21B2026
4	Jeevan Kumar Rajana	CS21B2041

RL Project Road Map

Problem Statement

This project investigates the effectiveness of Deep Reinforcement Learning (DRL) for Atari game playing by comparing the performance of Deep Q-Networks (DQN) and Asynchronous Advantage Actor-Critic (A3C) algorithms on the Breakout game. We aim to identify which approach yields superior performance and insights into their strengths and weaknesses in this specific environment.

Introduction

Achieving mastery in complex environments through Deep Reinforcement Learning (DRL) is a growing area of AI research. This project compares the performance of two prominent DRL algorithms, Deep Q-Networks (DQN) and Asynchronous Advantage Actor-Critic (A3C), in training an agent to play the classic Atari game Breakout. We will assess the effectiveness of each approach in achieving high scores and analyze their strengths and weaknesses within this specific game. This project aims to provide insights into the comparative performance of DQN and A3C for real-time strategy in Atari environments.

Approach

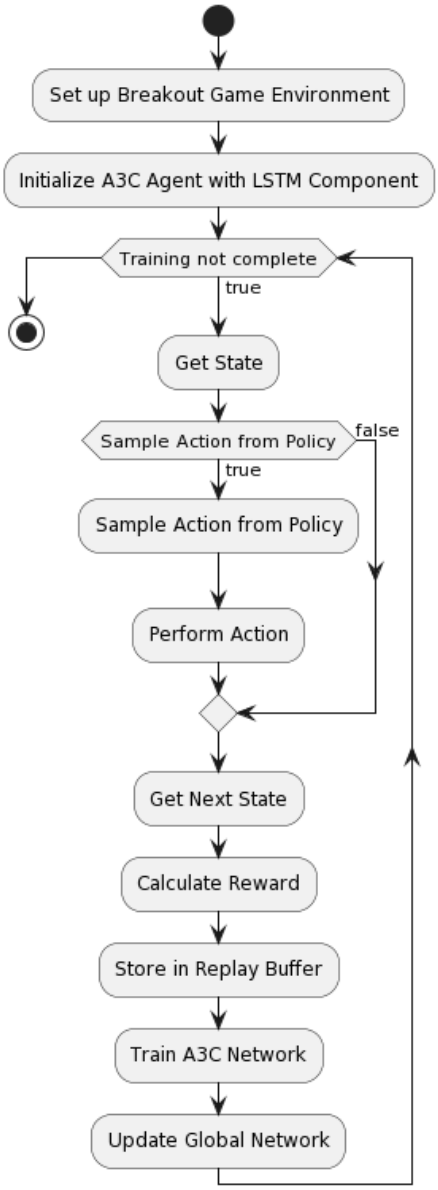
Approach for DQN:
1. Environment Setup: Set up the Atari Breakout game environment using the Pygame library, enabling the agent to interact with the game state and receive rewards based on its actions.
2. Deep Q-Network (DQN) Implementation: Implement the DQN algorithm using PyTorch, which approximates the Q-value function using a deep neural network.
3. Experience Replay: Utilize experience replay to store the agent's experiences (state, action, reward, next state) in a replay buffer, enabling efficient learning by sampling from this buffer during training.
4. Network Training: Train the DQN network by sampling experiences from the replay buffer and updating the network's weights to minimize the temporal difference error between the predicted and target Q-values.
5. Exploration-Exploitation: Adopt an exploration-exploitation strategy, such as epsilon-greedy, to balance exploration of new actions with exploitation of learned knowledge.
6. Evaluation: Evaluate the trained DQN agent by allowing it to play the Atari Breakout game and assessing its performance based on the achieved score or other relevant metrics.

Approach for A3C:
1. Environment Setup: Set up the Atari Breakout game environment, enabling the agent to interact with the game state and receive rewards based on its actions.
2. A3C Architecture: Implement the Asynchronous Advantage Actor-Critic (A3C) architecture, which uses multiple agents interacting with their own copy of the environment, improving diversity and efficiency.
3. Actor-Critic Model: Train an actor-critic model, where the actor learns the policy (action probabilities), and the critic learns the value function (state values), allowing for intelligent policy updates.

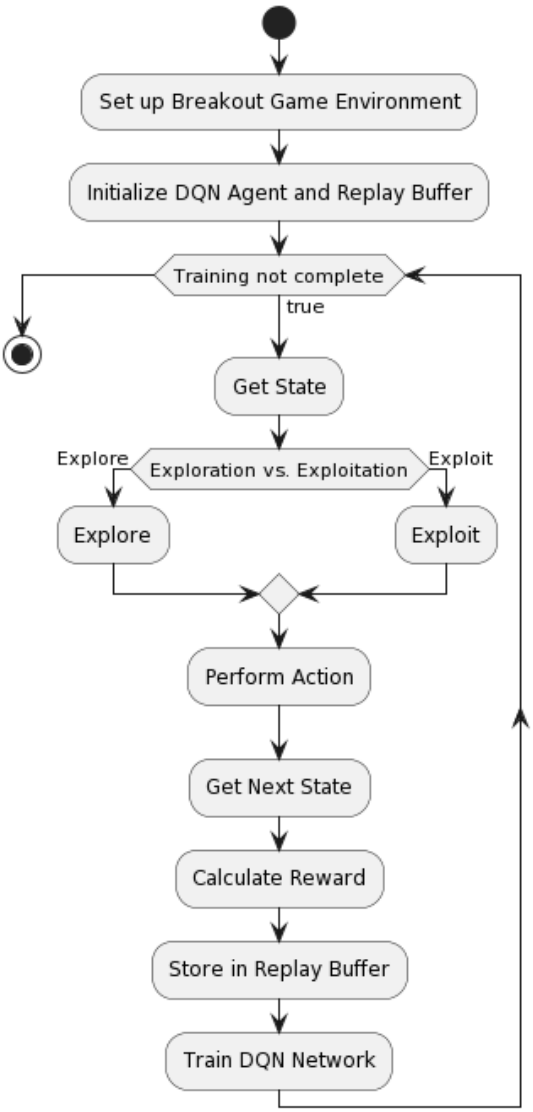
4. Advantage Estimation: Estimate the advantage function using the discounted returns and state values, focusing the agent's learning on actions that perform better than expected.
5. LSTM Integration: Incorporate an LSTM (Long Short-Term Memory) component to capture temporal dependencies and improve the agent's decision-making process.
6. Asynchronous Training: Train the global network asynchronously, updating it with the experiences from multiple agents in parallel, accelerating the learning process.
7. Evaluation: Evaluate the trained A3C + LSTM agent by allowing it to play the Atari Breakout game and assessing its performance based on achieved scores or other relevant metrics.

Overall Working Diagram

A3C Flowchart:-



DQN Flowchart:-



Design Details

Deep Q-Network (DQN) Approach:

1. Environment:

a. The Atari Breakout game environment, implemented using the Pygame library.

b. The environment provides a high-dimensional visual input (game screen) and a complex reward structure.

2. States: The state is represented by the current game screen, which is a 2D array of pixel values.

3. Actions: The agent can perform one of the following actions: move left, move right, or take no action.

4. Reward:

a. The agent receives a positive reward (+1) for each brick it breaks.

b. The agent receives a negative reward (-1) when the ball is missed, and the game ends.

5. Agent Definition (Mathematical Formulation):

a. The DQN agent aims to learn the optimal action-value function Q(s, a), which estimates the maximum expected future reward for taking action a in state s and following the optimal policy thereafter.

b. The Q-value function is approximated by a deep neural network, parameterized by weights θ:

$$Q(s, a; \theta) \approx Q^*(s, a)$$

c. The network is trained by minimizing the temporal difference (TD) error between the predicted Q-values and the target Q-values:

$$Loss = E[(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta))^2]$$

d. Experience replay is used to break the correlation between consecutive states and improve the efficiency of learning.

Justification:

a. DQN is a powerful reinforcement learning algorithm that can handle high-dimensional input states, making it suitable for the Atari Breakout game.

b. The use of a deep neural network allows the agent to learn complex mappings between the game screen and optimal actions.

c. Experience replay improves data efficiency and reduces the correlation between consecutive states, resulting in more stable learning.

d. The DQN approach has proven successful in various Atari games, as demonstrated in the seminal paper by Mnih et al. (2015).

Asynchronous Advantage Actor-Critic (A3C) Approach:

1. Environment: The Atari Breakout game environment, implemented using the Pygame library.

2. States:

a. The state is represented by the current game screen, which is a 2D array of pixel values.

b. Additionally, an LSTM (Long Short-Term Memory) component is used to capture temporal dependencies and improve decision-making.

3. Actions: The agent can perform one of the following actions: move left, move right, or take no action.

4. Reward:

a. The agent receives a positive reward (+1) for each brick it breaks.

b. The agent receives a negative reward (-1) when the ball is missed, and the game ends.

5. Agent Definition (Mathematical Formulation):

a. The A3C agent learns both a policy function $\pi(a|s; \theta)$ and a value function $V(s; \theta\_v)$, parameterized by weights θ and θ_v, respectively.

b. The policy function $\pi(a|s; \theta)$ outputs the probability distribution over actions for a given state s.

c. The value function $V(s; \theta\_v)$ estimates the expected future reward from state s, following the current policy.

d. The agent uses the advantage function $A(s, a) = R - V(s; \theta\_v)$ to update the policy, where R is the discounted future reward.

e. The policy and value functions are trained simultaneously using gradient descent on the policy loss and value loss, respectively.

f. Multiple agents interact with their own copies of the environment asynchronously, and their experiences are used to update a global network.

Justification:

a. A3C is an advanced reinforcement learning algorithm that combines the advantages of actor-critic methods and asynchronous parallel training.

b. The use of multiple agents interacting with their own environments improves the diversity of experiences and accelerates the learning process.

c. The actor-critic architecture allows for efficient policy updates based on the advantage function, leading to faster convergence.

d. The integration of an LSTM component helps capture temporal dependencies and improves decision-making in sequential environments like Atari games.

e. A3C has demonstrated state-of-the-art performance in various challenging environments, including Atari games, as shown in the original paper by Mnih et al. (2016).