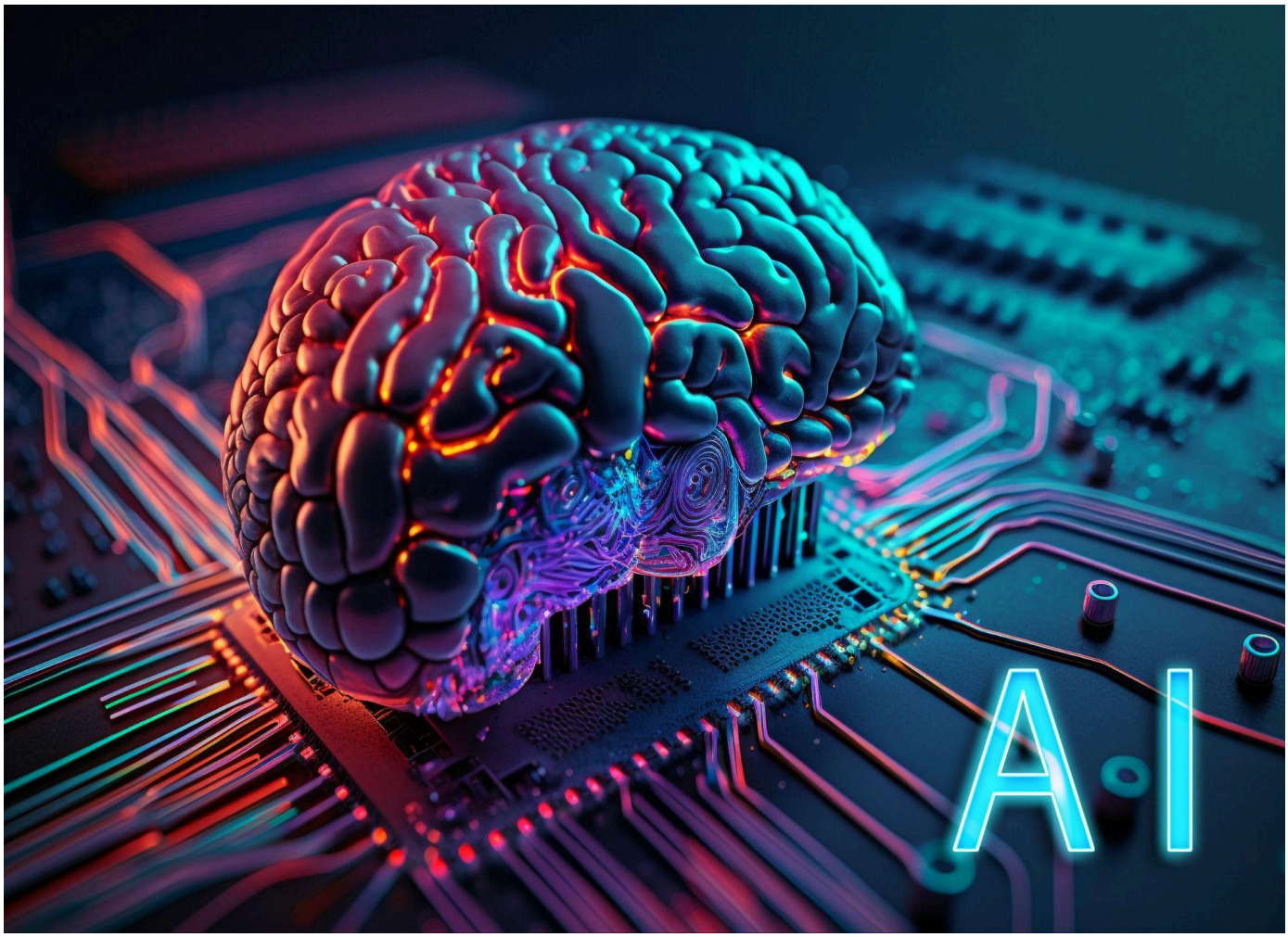
6FTC2086-0901-Social and Collective Artificial Intelligence

Practical Assignment

Omar Faisal Amr Mourad - 21026670



Introduction:

Objectives of the Project:

This project aims to implement a **Q-learning** algorithm in a grid environment where agents learn to navigate a 10x10 grid and reach a goal. The project includes single-agent and multi-agent simulations, focusing on:

- How agents learn to make decisions.
- Tracking their performance over time.
- Studying the impact of single agents vs. multiagent learning.

Q-Learning:

Q-learning is effective because the agent doesn't need to know anything about the environment beforehand. It learns by trying different actions and seeing what happens. Over time, the agent creates a **Q-table** that helps it figure out the best way to navigate the environment. By continuously updating the **Q-table**, the agent gets better at making decisions and learns to choose actions that give the highest rewards.

The main steps in Q-learning are:

1. The agent starts with an empty or initialized Q-table.
2. It takes an action and receives a reward based on the environment's feedback.
3. The Q-value for the state-action pair is updated using the **Q-learning formula**:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[R(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

- Q(s,a): Expected reward for taking an action (a) in state (s) - This represents the agent's knowledge about the value of that action.
- R(s,a): Immediate reward for taking action (a) in state (s) - For example, a positive reward for reaching the goal or a penalty for making a bad move.

- γ: Discount Factor - weighs future rewards - A value close to 1 means the agent considers future rewards, while a value near 0 focuses on present rewards.
- maxQ(s′,a′) - The best possible Q-value for the next state s′ - helps the agent consider the maximum reward it can achieve in the future.
- α: Learning rate - Controls how much the agent updates the Q-value based on new information - a higher value makes the agent learn faster, but too high a value may lead to instability.

To balance exploration and exploitation, I used the **epsilon-greedy method**:

- **Exploration**:
  - With a small probability (ϵ), the agent takes a random action. This allows the agent to explore new states and actions it hasn't encountered yet.
- **Exploitation**:
  - Most of the time (1−ϵ), the agent chooses the action with the highest Q-value for the current state, based on its current knowledge.

This approach ensures the agent doesn't get stuck in other paths and continues to learn effectively. Over time, ϵ can be gradually reduced (epsilon decay) so the agent focuses more on exploitation as it learns the defined policy.

Reinforcement Learning:

Reinforcement Learning (RL) is a learning method where agents improve their decision-making by interacting with an environment. The agent takes actions, observes the outcomes, and receives feedback in the form of rewards or penalties. Over time, this feedback helps the agent learn which actions lead to better results. RL is particularly useful when the environment is unknown or dynamic, as the agent learns through trial and error without needing prior knowledge.

Key components of RL:

- **States**: Represent the current situation of the environment. In Q-learning, the agent tracks each state and evaluates the rewards for taking actions in that state.
- **Actions**: The choices the agent can make in each state. Q-learning uses actions to explore the environment, updating its understanding of which actions are best through feedback.
- **Rewards**: Feedback from the environment indicating whether an action was good or bad. Q-learning uses rewards to update the Q-values in its Q-table, associating actions with their outcomes.
- **Policy**: The agent's strategy for selecting actions in each state to maximize rewards. In Q-learning, the policy evolves over time based on the Q-values, helping the agent choose actions that maximize its cumulative rewards.

Agent Behaviour:

Single-Agent Learning:

The single agent's learning performance improves steadily as it interacts with the grid environment. The agent starts with no knowledge of the grid or the optimal path to the goal, so it initially explores randomly, taking longer to reach the goal and earning lower rewards. Over time, as the agent updates its Q-table based on the rewards received, it starts exploiting its knowledge to take shorter and more efficient paths, resulting in higher rewards and shorter episode lengths.

Performance Over Time:

1. **Early Episodes**:
    - During the first 50–100 episodes, the agent heavily explores the grid.
    - Random exploration leads to inefficient movements and penalties for unnecessary steps, resulting in lower total rewards and longer episode lengths.

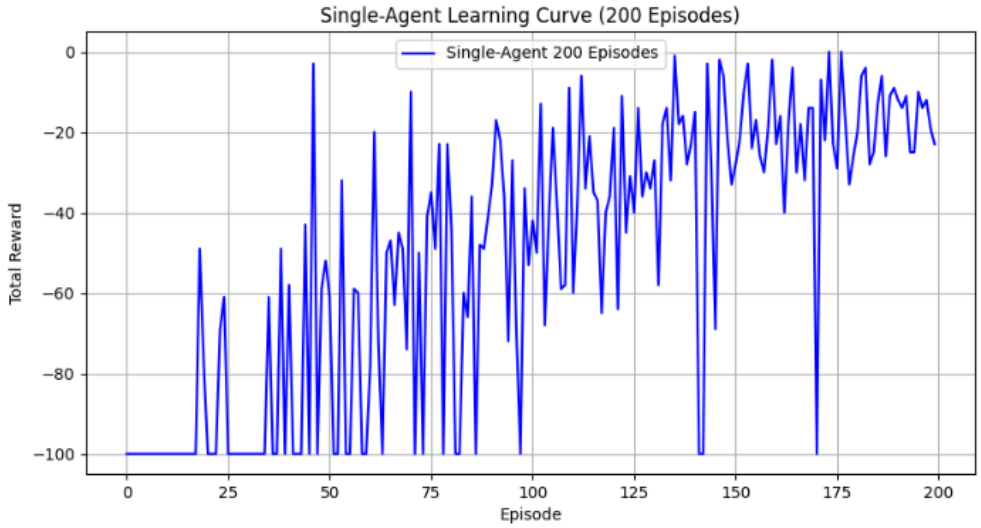
2. **Mid-Episodes (200–800)**:
    - The agent begins balancing exploration and exploitation, leveraging its Q-table to make better decisions.
    - Total rewards increase steadily, and episode lengths decrease as the agent learns to avoid obstacles and optimize its path to the goal.
3. **Late Episodes (800–1000)**:
    - By the 800th episode, the agent primarily exploits the optimal paths learned in its Q-table.
    - Episode lengths stabilize at their shortest possible value, and total rewards are consistently high, showing that the agent has fully learned the environment and its optimal strategy.

Learning Curve:

Below are the graphs showing the total rewards per episode for the single-agent simulation across 200, 800, and 1000 episodes.

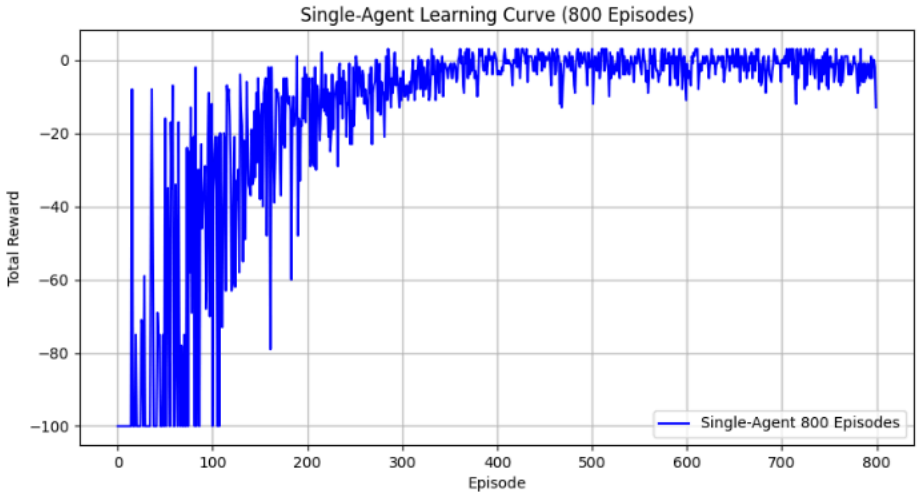
200-Episode Graph:

- In the 200-episode simulation, the agent is still in the early stages of learning. Most of its actions are exploratory, as it tries to understand the environment and build its Q-table.
- Rewards are inconsistent and generally lower because the agent frequently takes inefficient paths and incurs penalties for unnecessary moves.
- By the end of 200 episodes, the agent shows some improvement, with a gradual increase in rewards, but it has not yet fully exploited the optimal path to the goal. The learning process is still incomplete at this stage.



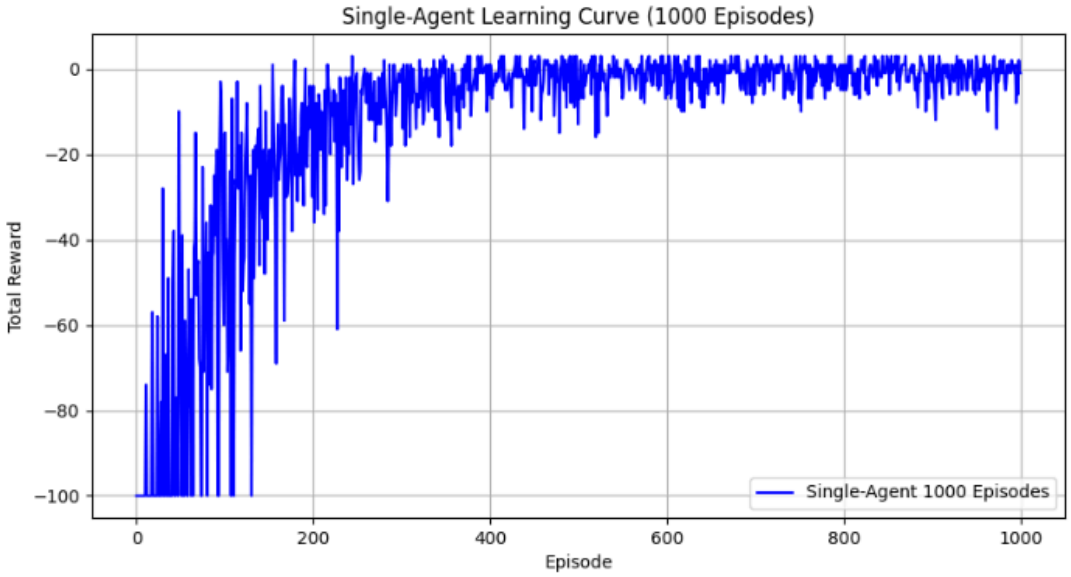
800-Episode Graph:

- In the 800-episode simulation, the agent demonstrates significant progress. Exploration decreases, and the agent begins exploiting its learned knowledge more effectively.
- Total rewards steadily increase, and the fluctuations seen in the 200-episode graph become less pronounced.
- By the 600th–800th episode range, the agent starts taking the shortest path to the goal in most episodes, leading to consistently higher rewards and shorter episode lengths. The graph reflects this improvement with a more stable upward trend.



1000-Episode Graph

- By the 1000-episode simulation, the agent has nearly completed its learning process. The rewards stabilize at their highest possible values, showing that the agent consistently uses the optimal path to the goal.
- Any remaining minor fluctuations in rewards are due to occasional exploration, which helps ensure that the agent's policy remains robust.
- The graph indicates convergence, with the agent having refined its Q-table to reliably select the best actions in all states.



The sudden drops in all three graphs occur due to the agent's exploration phase, where it sometimes selects suboptimal actions that lead to penalties or inefficient moves, causing negative rewards. Early in the training (especially in the 200-episode graph), the agent is still learning which actions lead to better outcomes, resulting in frequent drops in rewards as it experiments. Even in the 800 and 1000 episode graphs, occasional drops happen when the agent, despite having learned much, still makes less optimal decisions in some episodes. These drops are part of the trial-and-error process, with the agent gradually improving as it refines its Q-table and moves towards more consistent performance. The differences across these graphs highlight the agent's gradual improvement as it transitions

from exploration to exploitation. The longer the training, the better the agent performs, as reflected in the higher rewards and shorter episode lengths.

Multi-Agent Learning Behavior:

When multiple agents learn together in the grid environment, their performance is influenced by competition and independent learning. Each agent starts with no knowledge of the grid and builds its strategy based on rewards and penalties.

Learning Performance of Multiple Agents:

- **Early Episodes**:
    - At the start, agents explore randomly, often following overlapping paths to the goal. This competition slows progress, leading to longer episode lengths and lower rewards.
    - Their learning is inconsistent as they receive penalties for inefficient moves.
- **Mid Episodes (400–800)**:
    - As agents update their Q-tables, they begin to balance exploration and exploitation.
    - Some agents discover different routes to the goal, reducing competition. Rewards improve steadily, and episode lengths decrease as strategies become more efficient.
- **Late Episodes (800–1000)**:
    - By this stage, most agents have learned their optimal paths. They either find separate routes to the goal or adapt to share paths without much interference.
    - Rewards stabilize, and episode lengths are consistently short.

Collaboration and Competition:

- **Competition**:
    - In the early episodes, agents compete for the same goal, often following overlapping paths and slowing each other down.
- **Natural Collaboration**:
    - As agents learn independently, they adapt to avoid interfering with one another. While not necessarily working together, their strategies naturally spread out to reduce overlap.
- **Convergence**:
    - Over time, agents converge on efficient paths or find non-conflicting ways to reach the goal.

Learning Curve:

The graph below shows the total rewards for each agent over the episodes. It shows how their learning improves over time, with rewards stabilizing as agents converge on their strategies.



Effects of Parameters on the Learning Curve:

Epsilon (ϵ):

- A high epsilon at the start encourages agents to explore the grid, helping them learn about different paths and possibilities. However, this can lead to inefficient actions and slower initial progress.
- As epsilon decreases, agents shift to exploiting what they've learned, leading to faster and more consistent improvements in rewards.

Learning Rate (α):

- A high learning rate means agents adjust their Q-values quickly based on feedback. This helps them learn faster at the start but can cause instability if they overreact to single rewards.
- A low learning rate ensures stable updates, especially in later episodes, as the agent focuses on fine-tuning its learned strategy.

Discount Factor (γ):

- A high discount factor makes agents prioritize long-term rewards, helping them identify efficient paths to the goal.
- A low discount factor focuses on immediate rewards, which can lead to shorter but less optimal paths.

The same reason why there are sudden drops in the single agent graph applies here as well.

CSV Files Analysis:

single_agent_200.csv:

This csv contains the data from the single agent simulation over 200 episodes.

Strategy and Implementation:

- The agent starts from the position (0, 0) and moves through the grid until it reaches the goal at (9, 9) or reaches 100 steps.
- The CSV tracks **episode length** (the number of steps taken) and **total reward** (the sum of rewards during each episode), which can provide insight into how the agent improves its performance over time.

Observations:

- Early episodes will likely show longer episode lengths and lower total rewards because the agent is still exploring the environment, taking random actions, and over time learning the optimal path.
- There are some fluctuations in rewards as the agent learns the dynamics of its environment and improves its strategy.

single_agent_800.csv:

This file contains the data the single agent simulation over 800 episodes

Strategy and Implementation:

- As the agent explores less and exploits more, episode lengths are expected to decrease because the agent will be making better decisions, reaching the goal faster.
- The total rewards should increase as the agent is making more optimal decisions, taking fewer suboptimal or random actions.

Observations:

- There is a noticeable improvement in total rewards and episode lengths compared to the 200-episode simulation.
- The agent's performance has stabilized somewhat, with fewer fluctuations in rewards as it converges toward the optimal policy/path.

single_agent_1000.csv

This file captures the data for the single agent simulation after 1000 episodes, this is where the agent is expected to have fully learned the optimal strategy.

Strategy and Implementation:

- By this point, the agent should have almost completely transitioned to exploiting its learned Q-values, making optimal decisions and minimizing exploration.
- The agent's behavior is near-perfect, consistently choosing the best actions and reaching the goal in the fewest steps.

Observations:

- Total rewards are consistently high, as the agent is now effectively navigating the grid without wasting time on inefficient actions.
- The episode lengths should be shorter because the agent now takes the shortest possible path to the goal, without needing to explore suboptimal actions.

multi_agent_1000.csv:

This file records the data from the multi-agent simulation with the 4 agents, each trained for 1000 episodes. The multi-agent scenario shows how multiple agents interact within the same environment, each learning their own strategies for reaching the goal while considering their interactions with each other.

Strategy and Implementation:

- Each agent uses the same Q-learning approach but develops different strategies based on their individual experiences, since they might face different state-action-reward dynamics during training.
- The csv file tracks not just the total reward per episode but also the Q-values of each state-action pair.
- The environment consists of 100 states (10x10 grid), and agents perform actions which are Up, Down, Left, and Right to navigate the grid. The agents' positions and their Q-values are recorded, providing insights into how they learn to make better decisions.

Observations:

- Similar to the single-agent simulation, total rewards increase as the agents refine their strategies. However, multi-agent interactions introduce additional complexities as agents learn to navigate the grid in relation to each other's competition/collaboration.
- The Q-values for each state-action pair reveal the internal learning process of each agent and show how the agents converge towards optimal actions for each state.
- The episode length and total rewards show trends similar to the single-agent simulations, with agents gradually making more efficient decisions over time.

Conclusion:

This project showed how Q-learning works effectively for single-agent and multi-agent scenarios in a grid environment. The results show how agents learn good strategies over time by trial and error, slowly moving from random exploration to better decision-making. The single agent learned faster and more efficiently since it didn't face competition. It quickly found the shortest path, with its rewards stabilizing by the 800th episode. In multi-agent learning, early episodes were slower due to competition and overlapping strategies. However, as agents adapted, they learned separate paths or shared the environment more efficiently, performing similarly to the single agent by the 1000th episode.

Efficiency of the Learning Process:

- The single agent was more efficient because it had no competition, allowing it to focus entirely on optimizing its path.
- Multi-agent learning took longer to stabilize due to agents competing for the goal, but it showed that Q-learning can handle shared environments with multiple learners.
- The learning curves for both single and multi agents showed clear improvements in performance over time, showing the algorithm's ability to help agents converge on the best strategies despite early inefficiencies.

Potential Improvements:

- **Dynamic Parameter Adjustments**: Adjusting epsilon decay more carefully could help agents transition more smoothly from exploration to efficient goal-reaching, especially for multiple agents.
- **Collaborative Learning**: Adding mechanisms for agents to work together could reduce competition and help them find better strategies faster.
- **More Obstacles**: Having moving or randomly placed obstacles could create a more challenging environment, testing how well agents adapt.

In summary, Q-learning allowed both single and multiple agents to learn effective strategies. With these improvements, it could handle more complex scenarios and be more useful in real-world applications.