Final Project Report

TEAM-13

Shri Harsha Adapala Thirumala sadapala@buffalo.edu **50495139**

Maria Nivetha Antony Pushparaj marianiv@buffalo.edu **50550639**

Gayathri Sruthi Gannamaneni ggannama@buffalo.edu **50544968**

Grid world Co-operative Environment:

Our gridworld environment is comic inspired from the pages of the Batman comics. This grid world environment has 81 states, where every square represents a different location in the bustling city. Here, you'll encounter iconic characters like Batman, Robin, Catwoman (Selina Kyle), and infamous villains such as the Joker, Bane, and the Court of Owls.

The objective for Batman and Robin is clear: navigate the dangerous streets of Gotham without encountering any villains and reach Catwoman's location for a rewarding victory. Their movements are limited to up, down, left, or right, but they must exercise caution to avoid running into trouble with the likes of the Joker or Bane. Success means reaching Catwoman, but failure comes swiftly if they take too long or fall into the clutches of the villains one too many times.

In this multi-agent environment, both Batman and Robin must work together to achieve their goal. The environment tracks their every move, rewarding them for strategic movements and punishing them for moving far away from their goal state and make them follow the correct path.

To ensure the agent's safety and make the game versatile, the environment implements various strategies. Boundary checks are performed to keep the agents within the grid boundaries, preventing them from wandering outside the playable area. Also, if the agent stays in the same position even after performing action, it is given a penalty. Additionally, collision prevention measures are in place to avoid conflicts between Batman and Robin. Before updating their positions, the environment checks if their proposed movements coincide, ensuring that only one agent occupies a shared grid cell at a time. To conclude, this cooperative environment fosters collaboration between Batman and Robin as they strive to rescue Catwoman and emerge victorious against the forces of evil in Gotham City.

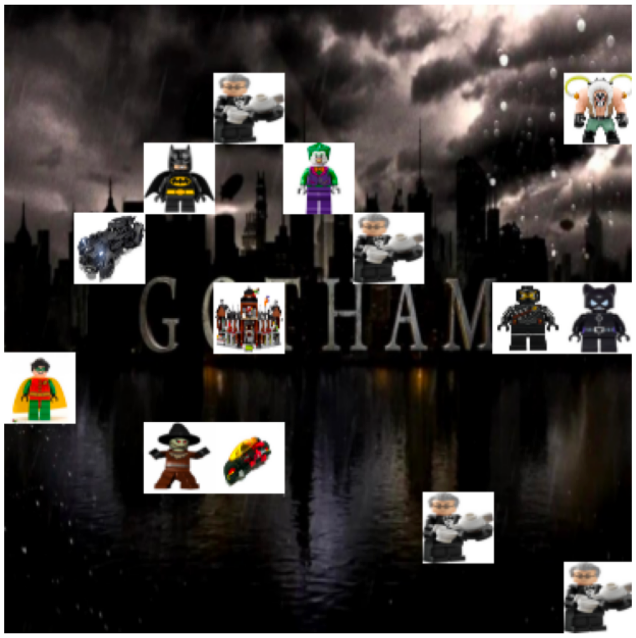
Grid World Characteristics and Components:

Characters:

- Batman – Agent 1
- Robin – Agent 2
- Selena – Goal State
- Batmobile – Reward
- Scarecrow – Obstacle
- Alfred - Obstacle
- Redbird - Reward
- Arkham - Obstacle
- Joker - Obstacle
- Court of owls - Obstacle
- Bane - Obstacle

Obstacles	Rewards
Selena	20
Joker	-10
Court of owls	-7
Bane	-5
Scarecrow	-3
Alfred	10
Arkham	-1 + Reset the agent to the original state
Batmobile	10 – for Batman
Redbird	10 – for Robin

Grid World Representation:



TD Methods:

Decentralized Q-learning:

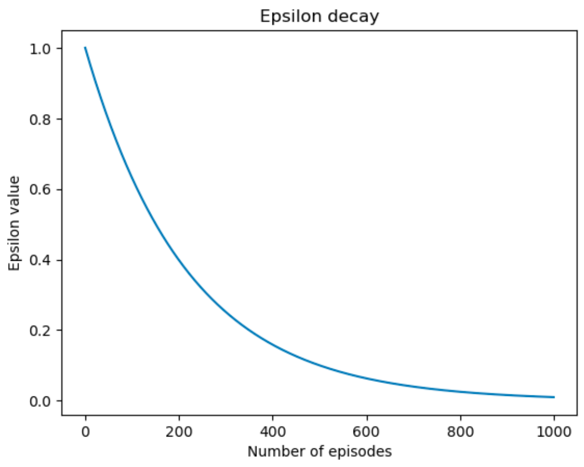
Decentralized Q-learning is a method employed in multi-agent reinforcement learning, wherein each agent independently learns from its interactions with the environment. The aim is to refine its own policy without direct coordination or communication with other agents. This approach proves beneficial in scenarios where agents possess diverse constraints or objectives, rendering centralized coordination impractical. Its key strengths lie in scalability and adaptability, rendering it applicable in both cooperative and competitive environments.

In practice, each agent maintains and updates its individual Q-table, leveraging rewards from individual actions to refine its policy and strike a balance between exploration and exploitation. While decentralized Q-learning proves advantageous in navigating complex systems involving multiple autonomous entities, such as robotics or distributed network management, it faces hurdles in ensuring optimal collective results.

However, the autonomy of agents introduces challenges, notably non-stationarity, wherein environmental dynamics unpredictably shift due to agents' simultaneous learning and actions. Despite its practical utility, decentralized Q-learning lacks robust theoretical convergence guarantees and grapples with issues like credit assignment and attaining optimal collective outcomes.

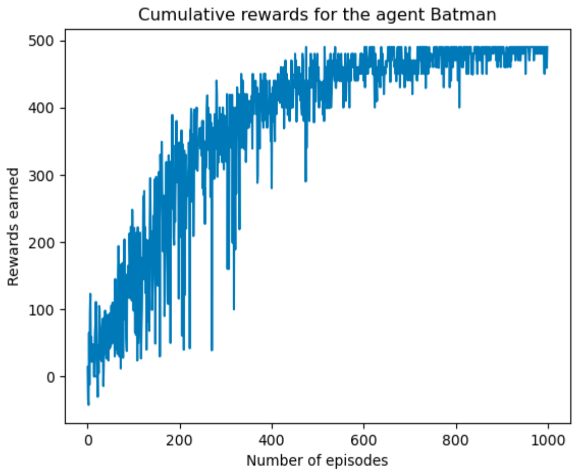
Performance of Decentralized Q-Learning in Grid World Environment:

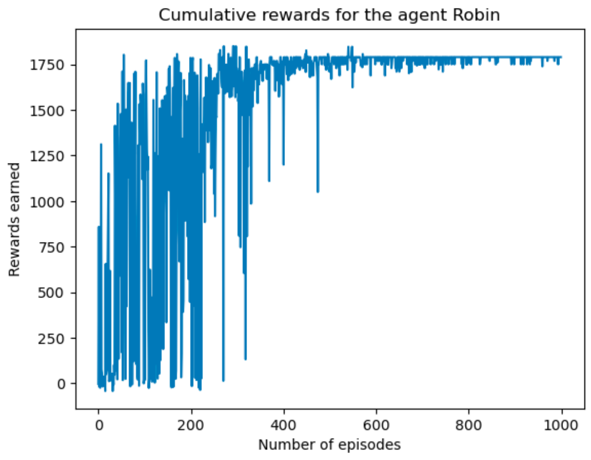
Epsilon Decay Graph:



We trained the agents for 1000 episodes to fill the Q-table and the epsilon value which was set from 1 reduced to near 0.

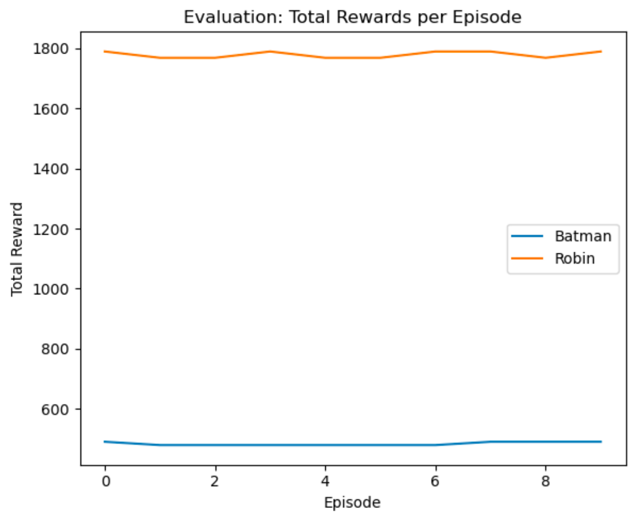
Cumulative rewards of agents during Training:

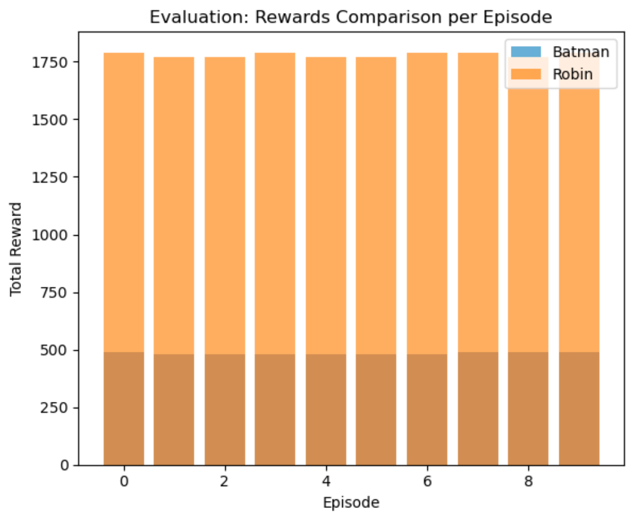




The cumulative rewards for the agent Batman and agent Robin almost converged after training 750 episodes. It is evident that agent Robin gains more rewards than agent Batman.

Cumulative rewards of agents during Testing:





These graphs for the agents Batman and Robin are relatively consistent and close together, which shows that both agents consistently receive similar rewards from episode to episode during evaluation. The bar chart compares the total rewards earned by Batman and Robin in each evaluation episode.

Correlated Q-learning:

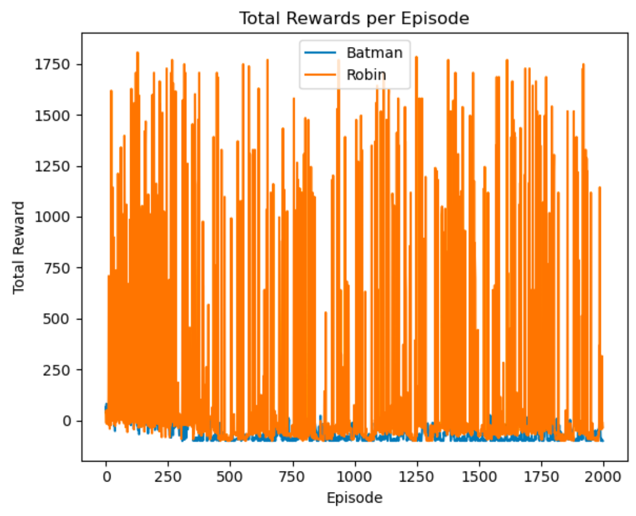
This approach not only holds promise for maximizing agent's expected returns but also streamlines computational efficiency in equilibria calculations through the integration of sophisticated linear programming techniques into its framework. By harnessing correlated Q-learning, agents can synchronize their actions effectively, leveraging joint action-value functions to cultivate cooperation and bolster decision-making within complex multi-agent systems.

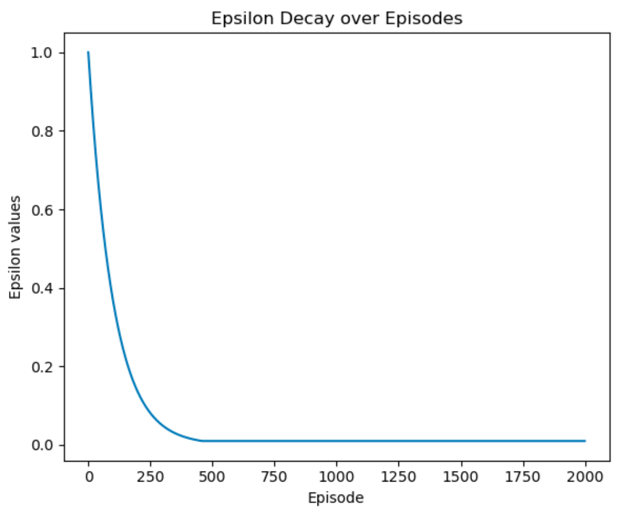
The implementation of correlated Q-learning involves a nuanced adaptation of the conventional Q-learning algorithm to accommodate the intricacies of correlated equilibrium computations. Rather than directly embedding a correlated equilibrium into each agent's strategy, this approach involves agents sampling joint actions from the equilibrium and subsequently adjusting their individual actions accordingly. This strategic maneuvering serves to uphold the crucial correlations among agents' actions, fostering a harmonized approach to achieving collective objectives.

However, while correlated Q-learning presents a promising avenue for enhancing cooperative behaviors among agents, the task of selecting the most suitable equilibrium remains a formidable challenge. This challenge stems from the expansive array of potential solutions inherent in correlated equilibria, which surpasses the comparatively constrained landscape of Nash equilibria. Consequently, navigating the complexities of correlated equilibria demands careful consideration and robust analytical techniques to ensure optimal decision-making outcomes.

Performance of Correlated Q-Learning in Grid World Environment:

Epsilon Decay and Cumulative rewards of agents during training:

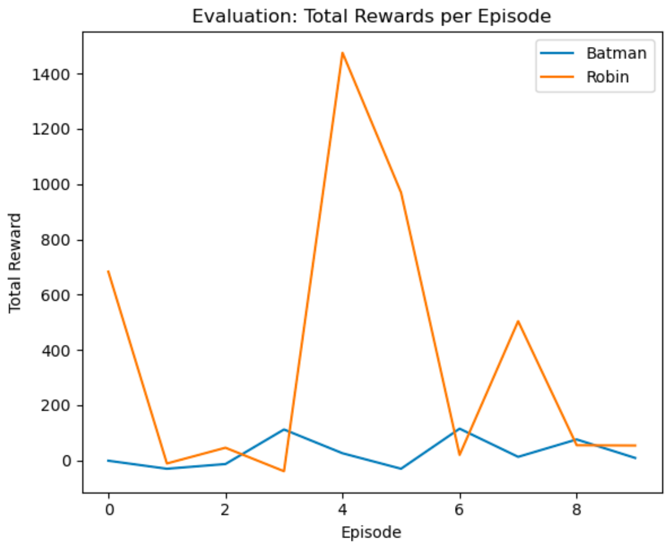


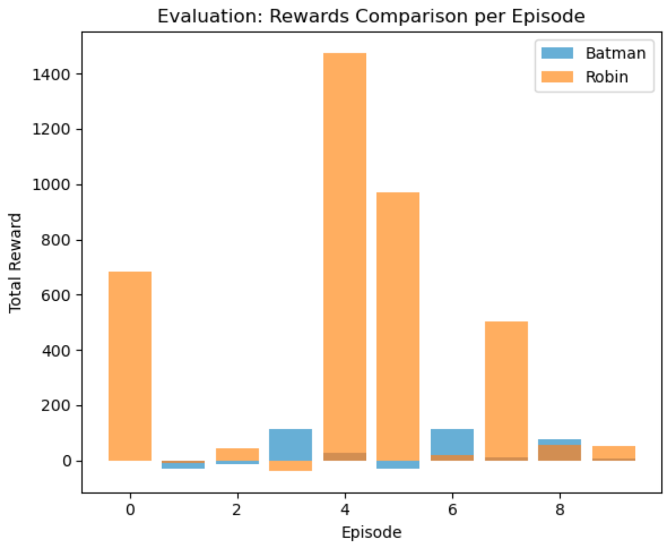


The graph shows the total rewards earned by the agents, Batman and Robin, in every episode of their training in Correlated Q learning.The agent Robin's rewards are very high and fluctuate significantly, suggesting Robin's policy may involve taking high-risk actions that can lead to both high rewards and potential losses. The other agent Batman's rewards are much lower, which may suggest a more conservative strategy or a less effective policy. The sharp peaks and troughs in Robin's rewards indicate episodes with varying outcomes, which could be due to the exploratory nature of the learning algorithm, the stochasticity of the environment, or interactions with Batman's policy.

In the same way, epsilon is high during the initial phase of training, allowing for greater exploration of the state-action space. As learning progresses, epsilon decreases, indicating a shift towards exploiting the learned policies.

Cumulative rewards of agents during Testing:





We trained the agents for about 2000 episodes and obtained these results. This suggests that it requires some more hyperparameter tuning to obtain better results.

REINFORCE:

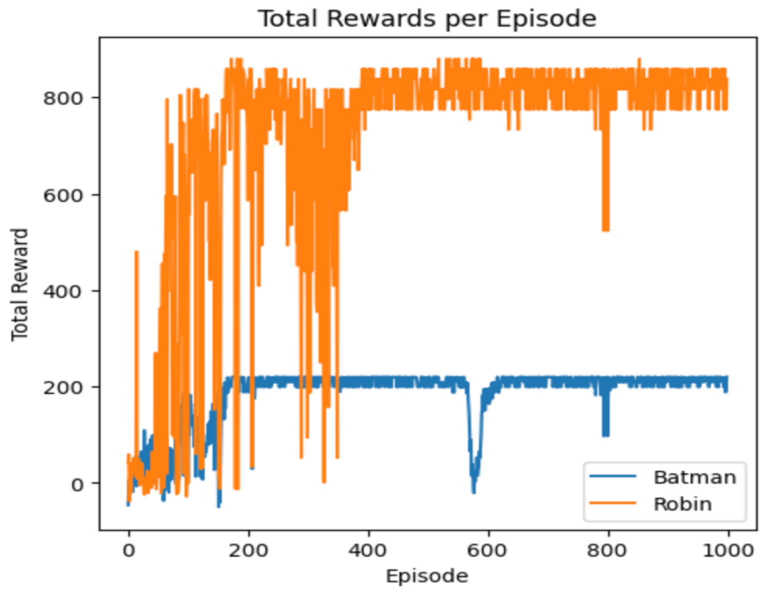
The policy gradient theorem, fundamental in reinforcement learning (RL), extends its significance to multi-agent reinforcement learning (MARL). In MARL, individual agents optimize their policies independently to maximize returns. The REINFORCE algorithm, a manifestation of this theorem in MARL, iteratively refines policies by leveraging Monte Carlo return estimates computed from full episodic trajectories. However, REINFORCE encounters challenges due to high variance in Monte Carlo return estimates, leading to unstable training.

To address high variance, practitioners commonly employ baselines with REINFORCE. By subtracting a baseline, typically the state-value function, variance of gradients is reduced while maintaining policy optimization towards higher expected returns. Actor-critic algorithms, a prominent family of policy gradient methods in MARL, train both a parameterized policy (actor) and a value function (critic) concurrently, enhancing stability and convergence.

Balancing the trade-off between bias and variance, actor-critic algorithms offer stability and efficiency in training policies within MARL. Notable examples include advantage actor-critic (A2C) and proximal policy optimization (PPO), each offering unique advantages in training efficiency and convergence properties. These algorithms play a crucial role in advancing MARL systems, enabling agents to learn effective policies in dynamic multi-agent environments.

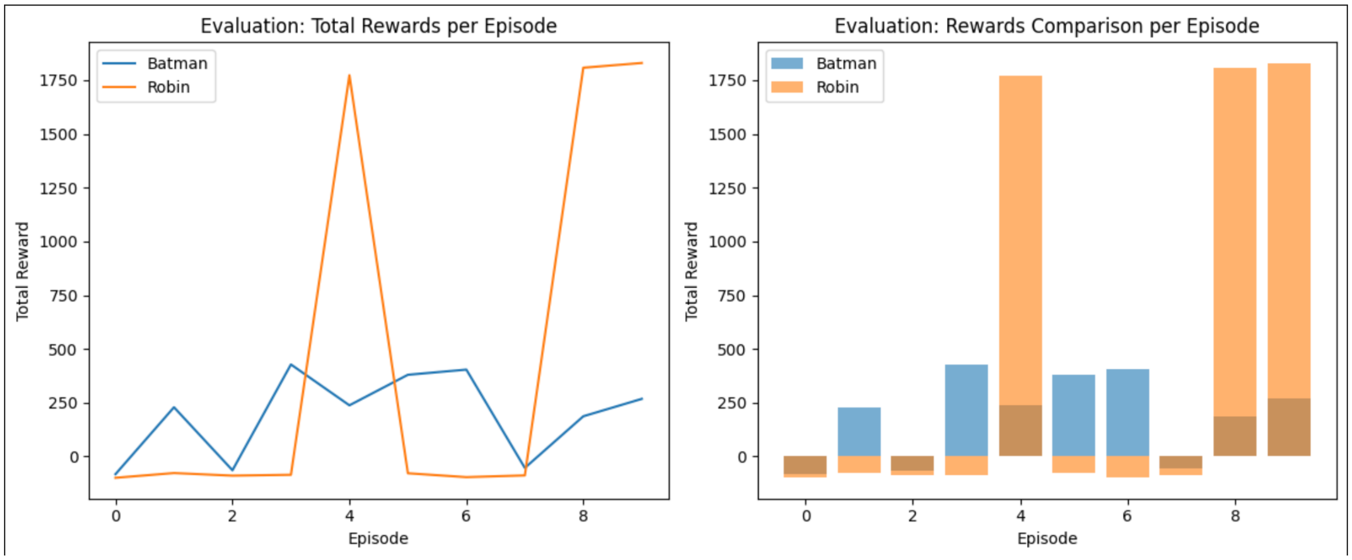
Performance of REINFORCE in Grid World Environment:

Cumulative rewards of agents during Training:



The graph compares the performance of two agents, Batman and Robin, across 1000 episodes. Robin's performance is consistently high, with rewards around 800, apart from occasional drops, indicating a generally effective policy with sporadic lapses possibly due to complex challenges and policy effects. In stark contrast, Batman's rewards are highly erratic at the start and almost near to zero around the 180th episode and then receives a steady reward of 200 which suggests a robust policy.

Cumulative rewards of agents during Testing:



Batman's Reward exhibits some variability, beginning with modest rewards and reaching noticeable peaks around episodes 3 and 6. The trend shows a gradual increase in rewards as the episodes progress, particularly towards the final episodes depicted. Robin's Reward is characterized by significant fluctuations. Notably, there is a sharp peak at episode 6, where Robin achieves a very high reward, indicative of a particularly successful episode. The reward pattern also shows a peak in the earlier episodes and maintains a level of variability across the series. The chart clearly shows that Robin generally earns more rewards than Batman, except in episodes where both seem to have minimal rewards.

REINFORCE algorithm is not suitable for this grid world environment but maybe with proper hyperparameter tuning can achieve expected results.

MADDQN:

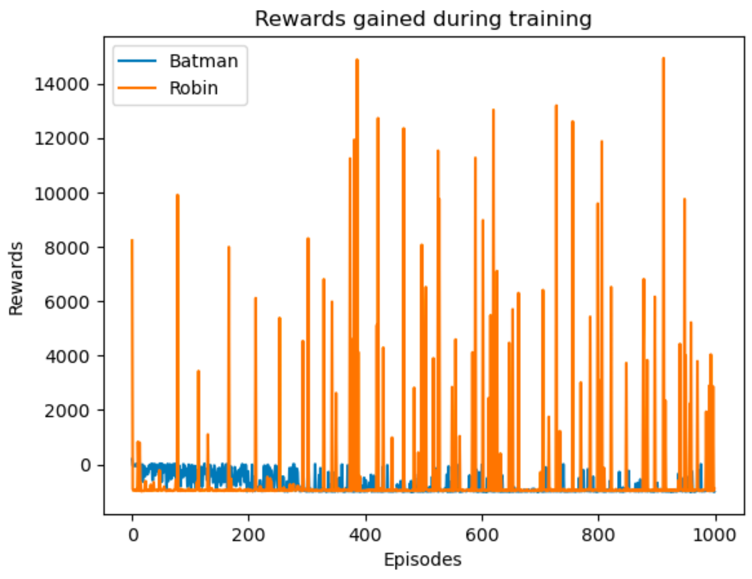
Multi-Agent Deep Double Q-Network (MAD-DQN) represents a significant stride forward in multi-agent reinforcement learning (MARL), building upon the foundations set by the Deep Q-Network (DQN) algorithm. In MARL, agents grapple with navigating intricate environments, engaging with one another, and making decisions based on their observations and the actions of fellow agents. MAD-DQN extends the DQN framework to accommodate multiple agents, empowering them to develop decentralized policies autonomously while considering the actions and strategies of their peers. This setup fosters more efficient exploration and collaboration among agents across diverse cooperative or competitive scenarios.

An important aspect of MAD-DQN lies in its utilization of a decentralized architecture combined with a double Q-learning methodology. Each agent in MAD-DQN manages its own ensemble of Q-networks, comprising both target and local networks. These networks enable agents to independently gauge the value of their actions in various states, thus mitigating the problem of overestimation frequently encountered in conventional Q-learning techniques. Through the incorporation of double Q-learning, MAD-DQN enhances the reliability and convergence of learning, resulting in more resilient and effective policies within complex multi-agent environments.

Moreover, MAD-DQN leverages techniques like experience replay and target network updates to augment learning efficiency and stability. Each agent maintains its own reservoir of past experiences, storing interactions encountered during engagements with the environment. During the training process, agents draw upon these experiences to refine their Q-networks, facilitating more informed learning from historical encounters and breaking temporal dependencies between successive experiences. Additionally, periodic adjustments to the target Q-networks serve to maintain stability by reducing variations in Q-value estimates over time. In essence, MAD-DQN offers a potent framework for nurturing decentralized policies in multi-agent environments, fostering enhanced cooperation and competition among agents while grappling with prominent challenges in MARL.

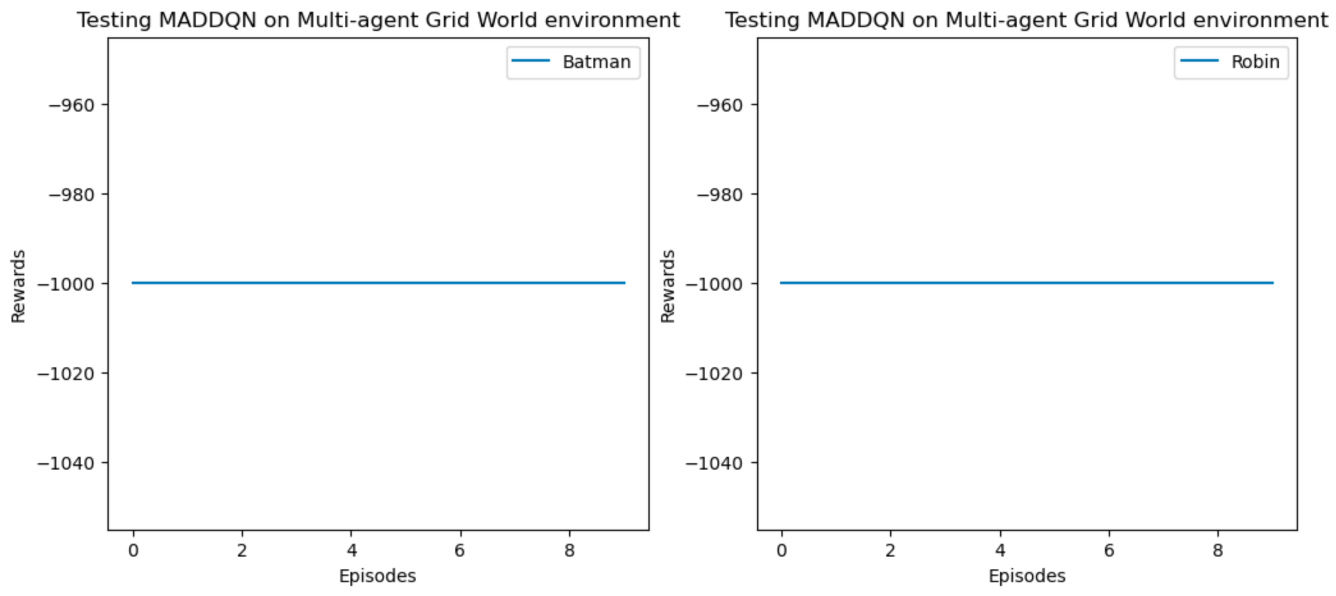
Performance of MADDQN in Grid World Environment:

Cumulative rewards of agents during Training:



When training the model for about 1000 episodes, robin's performance line shows a significant variability throughout the training episodes. The rewards range from 2,000 to nearly 14,000 with numerous peaks suggesting episodes of high performance. The distribution and frequency of the graph indicates that even while Robin's performance is fluctuating, there are many episodes where Robin is achieving high rewards.
On the other hand, Batman's performance is consistently low initially, hovering close to zero throughout the 1,000 episodes. Then the performance of Batman improves over time and actively engages in a way that earns rewards within the training environment.

Cumulative rewards of agents during Testing:



The Batman’s Rewards graph displays the rewards that Batman received across 10 episodes. The rewards are consistently around –1000. This suggests Batman may have effectively learned the optimal strategies and behaviors necessary for the environment.

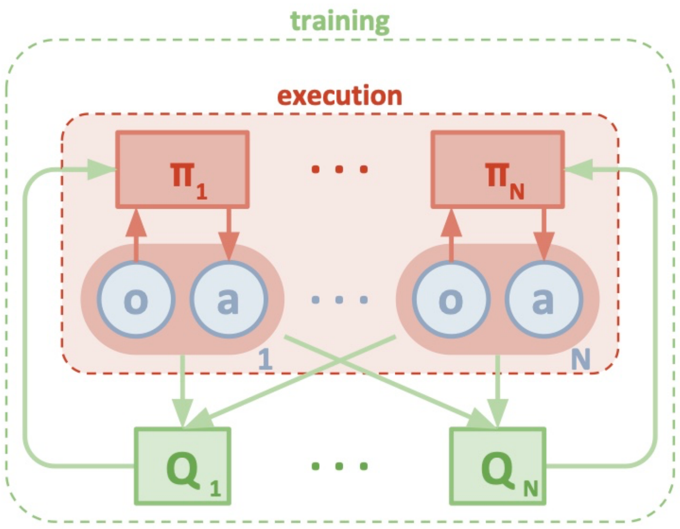
The Robin’s Rewards graph shows the rewards for Robin over 10 episodes. Robin’s rewards are also consistently stable at around –1000 which suggests a similar pattern with Batman’s performance, suggesting Robin also struggled to achieve positive outcomes.

Multi-Particle Environment (Multi Agent Mixed Cooperative-Competitive):

The Simple Adversary environment is designed for the exploration of adversarial interactions in reinforcement learning, focusing on the dynamics between a primary agent and its opponent. In this environment, the primary agent is tasked with achieving a specific goal or reaching a designated state within the environment and the opposing agent's mission is to prevent this success. Both agents use reinforcement learning techniques while adjusting their actions based on the rewards and penalties received, which allows them to continuously evaluate and refine their approaches. This environment's design aids in dissecting the complex behaviors that emerge during adversarial conflicts, making it easier to study and develop strategies. This environment is mostly for testing algorithms destined for more intricate applications such as systems requiring robust defense mechanisms and competitive scenarios where each agent's objectives are mutually exclusive. This environment is crucial for advancing our understanding of how agents can navigate and succeed in environments characterized by opposition and conflict.

Algorithm:

We are implementing **Multi-Agent Deep Deterministic Policy Gradient Algorithm** with a knowledge sharing framework of **Centralized Learning and Decentralized Execution**.



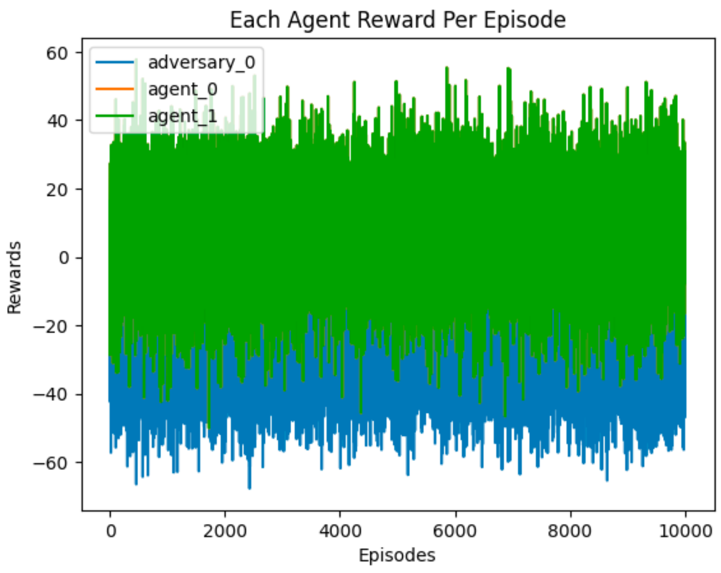
We have three distinct agents: adversary-01, agent-01, and agent-02. Each agent operates within its own network architecture, comprising both actor and critic networks. This implies that for the entire MADDPG algorithm, we require a total of six networks, with each agent possessing its own dedicated actor and critic networks. The actor network for each agent dictates its policy, guiding its actions within the environment autonomously, without dependence on the policies of other agents. On the other hand, the critic network for each agent functions differently; it has access to the critic networks of all agents, including its own, enabling it to assess the values associated with the actions taken by each agent within the collective environment. MADDPG uses OFF-Policy for critic and ON-Policy for actor.

Centralized Learning and Decentralized Execution serve as guiding principles within MADDPG. During the learning phase, the critic networks play a pivotal role by providing comprehensive insights into the actions and their respective values across all agents. This centralized learning approach allows for a holistic understanding of the environment, facilitating the refinement of policies that are not only beneficial to individual agents but also conducive to collective success. However, during execution, each agent operates independently, relying solely on its own actor network to make decisions based on local observations. This decentralized execution ensures agility and efficiency in real-time decision-making, enabling agents to adapt swiftly to dynamic environmental changes without the need for coordination with other agents.

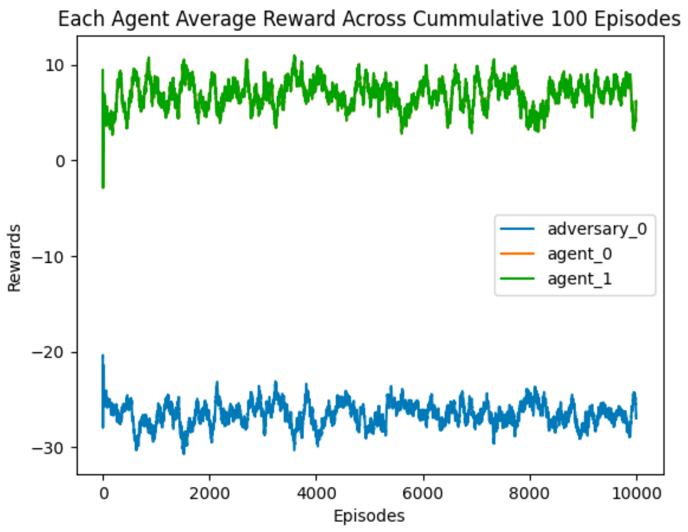
The MADDPG framework embodies a delicate balance between centralized learning and decentralized execution, leveraging the collective intelligence of all agents while preserving their autonomy during action selection. By combining insights from both centralized and decentralized perspectives, MADDPG equips agents with the capabilities to learn effective strategies collaboratively while executing decisions autonomously. This approach fosters synergy among agents, allowing them to navigate complex multi-agent environments adeptly and achieve collective goals efficiently.

Performance of MADDPG algorithm in MPE Simple Adversary Environment:

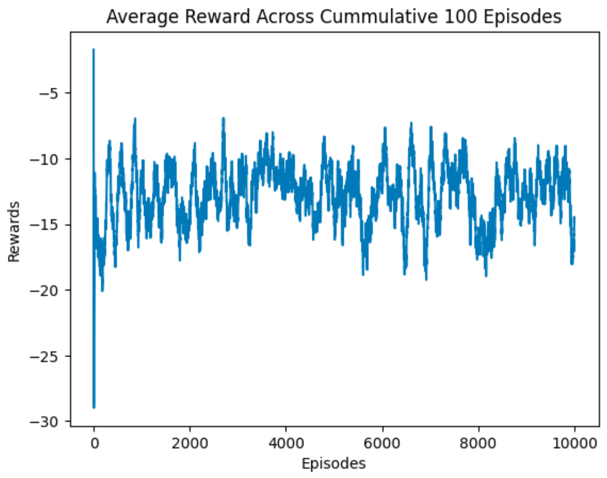
Cumulative rewards of agents during Training:

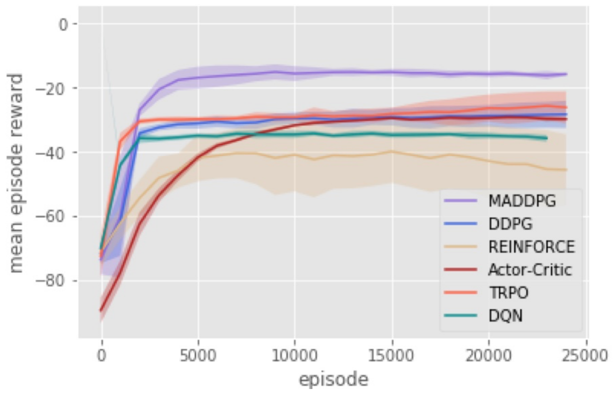


This is the cumulative rewards of the two agents during the training process we conducted for 10000 episodes. Since the agent_0 behaves like the agent_1, its rewards are not evident in the graph. Also, since this is a game, the average reward should be around 0.



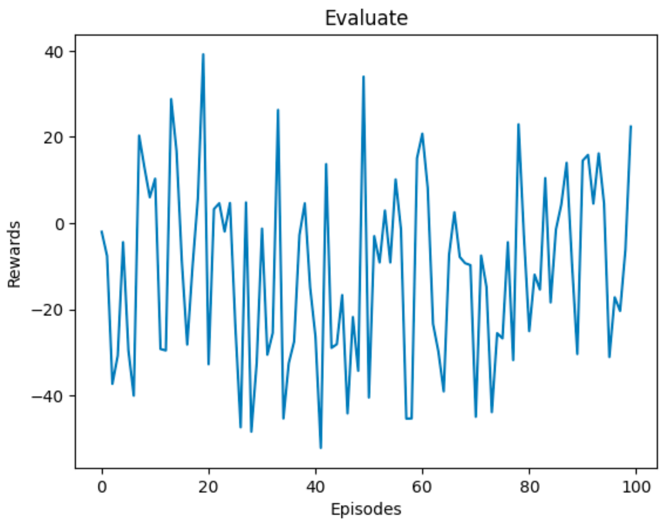
The rewards for these two agents are symmetrical. They contrast with each other, where if one agent tends to go upwards but the rewards of others move downwards.





The average rewards noted for the trained 10000 episodes is in the range of -15 to –10. The rewards are consistent after some initial fluctuations throughout the end. The right graph is based on the paper[5]. Comparing the performance from the original paper, our model works like that of the proposed network.

Cumulative rewards of agents during Testing:



This graph reflects the trained model's performance during evaluation. Here, it has an average reward of 0, which a simple game should follow.

Contribution:

Team Member	Project Part	Contribution
Maria Nivetha Antony Pushparaj	Grid world environment	33.33
Shri Harsha Adapala Thirumala	Grid world environment, Complex environment	33.33
Gayathri Sruthi Gannamaneni	Grid world environment	33.33

References:

1. https://openreview.net/pdf?id=xdZs1kf-va
2. Multi-Agent Reinforcement Learning Book by Stefano V. Albrecht, Filippos Christianos, and Lukas Schäfer
3. https://towardsdatascience.com/decentralized-reinforcement-learning-eefb718f2e34
4. MARL Environment Demo by Professor Alina Vereschaka
5. MADDPG for Competitive and Cooperative Mixed Environment: https://arxiv.org/pdf/1706.02275