Multi-Agent Reinforcement Learning for Drone Delivery Coordination and Simple Spread (v3) from Petting Zoo

CSE 4/546: Reinforcement Learning, Spring 2025

TEAM 25
- Aishwarya Virigineni - 50595813
- Prajesh Vizzapu - 50602532
- Nithya Kaandru - 50604120

University at Buffalo
School of Engineering and Applied Sciences

Project Description

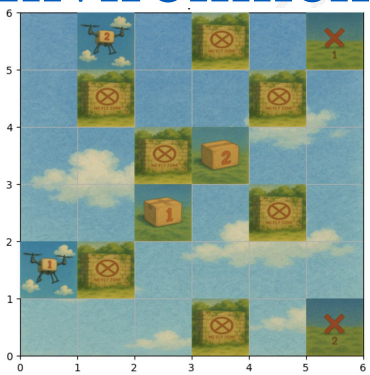
- Our project investigates how Multi-Agent Reinforcement Learning (MARL) enables coordinated decision-making in complex environments with multiple autonomous agents.
- We explore two distinct scenarios:
  a) A custom drone delivery system, where drones must pick up and deliver packages across a city grid, navigating no-fly zones, avoiding collisions, and optimizing delivery efficiency.
  b) Simple Spread v3 (Petting Zoo), a cooperative MARL benchmark where agents aim to cover all landmarks while minimizing overlap and collisions.
- Agents are trained to develop cooperative strategies in partially observable settings using both tabular methods (Q-Learning, SARSA, Double Q) and deep reinforcement learning algorithms (DQN, QMIX, DQN, Double DQN, Dueling DQN).
- The environments reflect real-world challenges in urban air mobility, warehouse automation, and decentralized multi-agent systems.

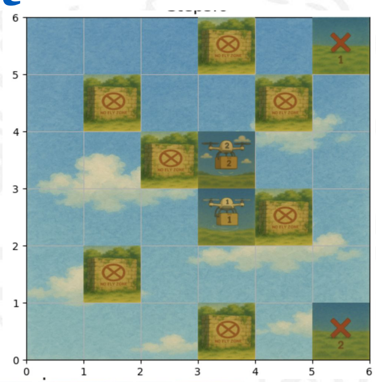
Multi Agent Reinforcement Learning

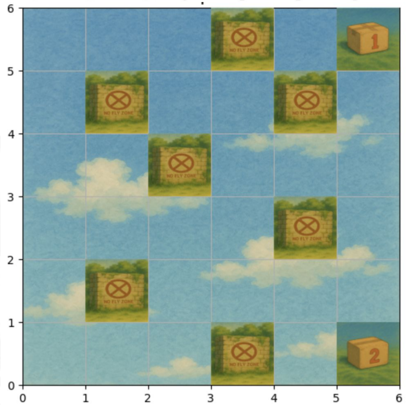
- Multi-Agent Reinforcement Learning (MARL) involves training multiple agents to interact in a shared environment, where outcomes depend on collective actions.
- Learning Paradigms in MARL:
  a) **Independent Learning:** Agents learn their own policies, treating others as part of the environment (e.g., Independent Q-Learning).
  b) **Centralized Training, Decentralized Execution (CTDE):** Uses shared information during training (e.g., joint action-value functions) but allows agents to act independently during execution (e.g., QMIX, MADDPG).
- Challenges in MARL: Non-stationarity, Credit Assignment, Scalability, Partial Observability

Drone Delivery - Custom MARL Environment

- **Grid Size**: 6 × 6
- **Agents**: 2 drones (drone_1 , drone_2)
- **Packages**: 2 (with unique pickup & drop-off locations)
- **No-Fly Zones**: Restricted areas with penalty
- **Stochastic Mode**: Optional random action noise (10% chance)
- **Action Space (Discrete: 6)** : Up, Left, Right, Down, Pick up, Drop Off
- **Observation Space (Tuple)**: (x position, y position, is_carrying [0 or 1])
- **Rewards**: +25 for pickup, +100 for delivery, -1 per step, -10 for collisions, -25 for invalid actions, -50 for hitting borders, -100 for no-fly zone.
- **Visual Rendering:** Uses custom images for drones, packages, drop-offs, and background & Supports real-time grid visualization.





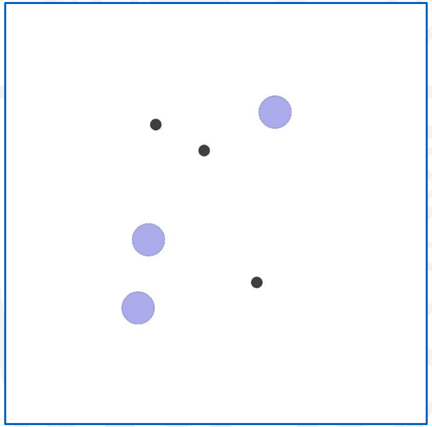


MARL Support in Drone Delivery Environment

- **Multi-Agent Framework**
  - a) **Agent Setup:** Flexible drone count using agent_ids
  - b) **Dict-Based Interface:** step() and observations support multi-agent training
  - c) **Centralized State Access:** Full environment snapshot via get_centralized_state()
- **Cooperative Learning Design**
  - **a)** Shared Reward across agents to promote teamwork
  - **b)** Coordination required for:
    - ➢ Package pickups (avoid duplicates)
    - ➢ Drop-offs (correct location)
    - ➢ Collision avoidance
  - **c)** Supports both decentralized policies and centralized critics
- **Real World Concepts Simulated**
  - ➢ Multi-agent logistics & navigation
  - ➢ Shared constraints (no-fly zones, collisions)
  - ➢ Reward-based decision-making under partial observability

Simple Spread V3- Petting Zoo - MARL Environment

- **Scenario:** Cooperative multi-agent task where agents must spread out and cover landmarks
- **Agents:** 3 agents (agent_0, agent_1, agent_2)
- **Landmarks:** 3 static targets to be covered
- **Objective:** Minimize total distance between agents and landmarks
- **Action Space:** Discrete (5) – [no-op, left, right, down, up]
- **Observation Space:** Continuous vector with
  - **a)** Own velocity & position
  - **b)** Relative positions of landmarks and other agents
- **Rewards**
  - **a) Global**: Negative sum of distances from landmarks to nearest agents
  - **b) Local**: -1 penalty per agent collision



https://pettingzoo.farama.org/environments/mpe/simple_spread/

MARL Support in Simple Spread V3

- **Multi-Agent Framework**

  a) Agents alternate actions in AECEnv (agent-environment cycle)

  b) Also available as parallel_env() for simultaneous action interface

  c) Each agent receives its own observation, acts individually

- **Cooperative Learning Design**

  a) Shared Reward

  - ➢ Negative sum of distances from each landmark to closest agent
  - ➢ Promotes teamwork and optimal landmark coverage

  b) No individual agent rewards

Methods

Drone Delivery Environment

- **Q-Learning:** Learned optimal policies via value iteration.
- **SARSA:** Trained agents using on-policy updates to adapt to environment dynamics.
- **Double Q-Learning:** Reduced overestimation bias by decoupling action selection and evaluation.
- **DQN:** Used neural networks to approximate Q-values.
- **QMIX:** Enabled centralized training with decentralized execution using a mixing network for joint action-values.

Simple Spread v3 (PettingZoo)

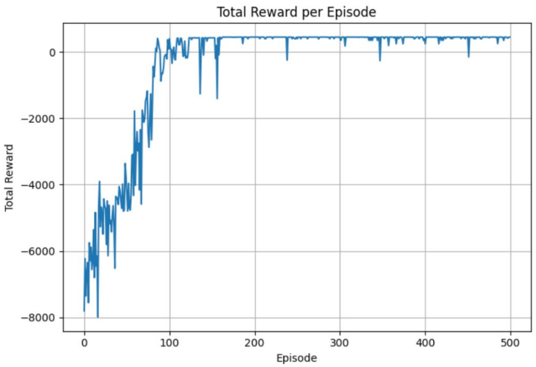
- **DQN:** Applied deep Q-networks for discrete cooperative action learning.
- **Double DQN:** Improved stability by mitigating Q-value overestimation.
- **Dueling DQN:** Separated state value and advantage estimation to enhance learning efficiency.

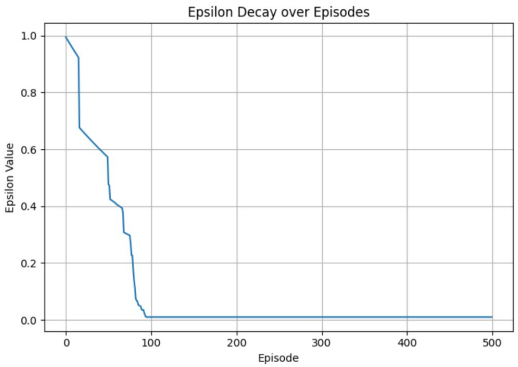
RESULTS - DRONE DELIVERY SYSTEM

A. Using Tabular Methods:

1. Q-LEARNING

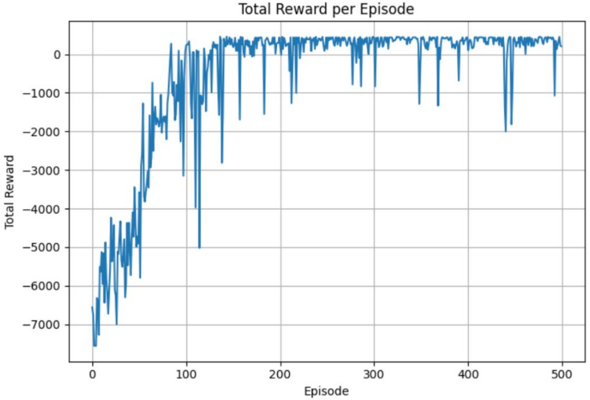
- **Reward Progression:**
  Sharp increase in rewards with rapid convergence by around episode 120.
- **Epsilon Decay:**
  Fast decay from 1.0 to near-zero within 100 episodes, leading to quick exploitation.
- **Greedy Evaluation:**
  Greedy policy yields high rewards (~450), with negligible variance.
- **Stability:**
  Post-convergence rewards remain consistently high with minimal drops.
- **Summary Insight:**
  Q-Learning delivers fast and effective training with strong test-time performance.

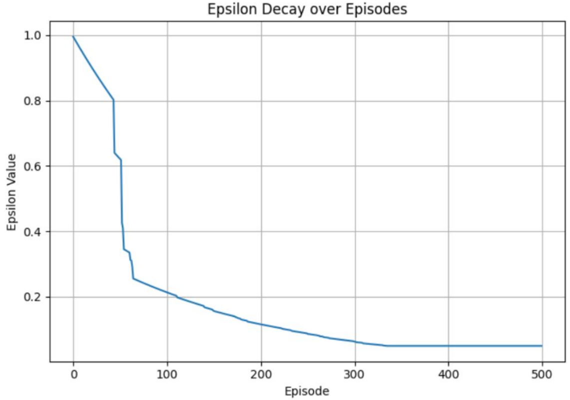




2. SARSA

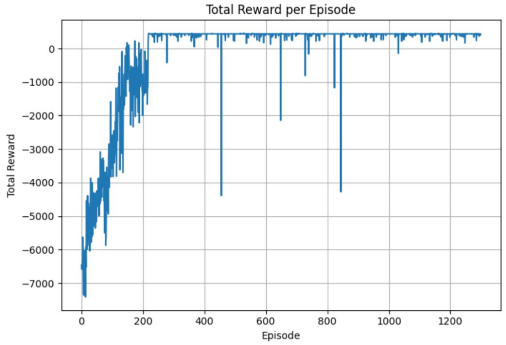
- **Reward Progression:**
 Steady and gradual improvement in total reward after initial fluctuations.
- **Epsilon Decay:**
 Smooth and slower epsilon decay curve compared to others, promoting longer exploration.
- **Greedy Evaluation:**
 Greedy policy rewards stabilize around 438, showing consistent test-time performance.
- **Stability:**
 Training is stable with minimal reward spikes after convergence.
- **Summary Insight:**
 SARSA offers robust and reliable convergence, ideal for stable environments.

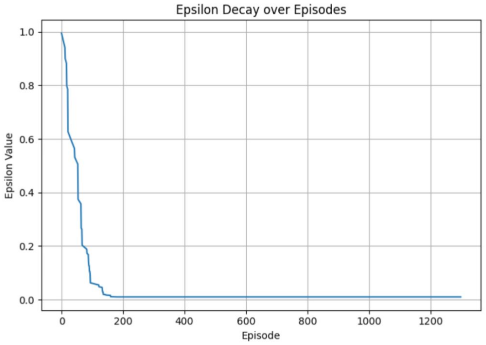




3. DOUBLE Q-LEARNING

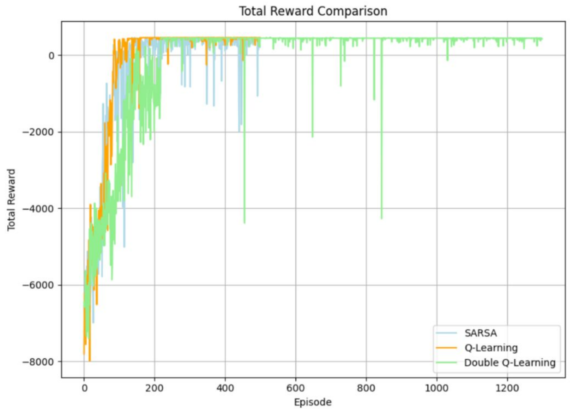
- **Reward Progression:**
  Slower convergence compared to Q-Learning, with more fluctuations across episodes.
- **Epsilon Decay:**
  Epsilon drops quickly by episode 200, allowing controlled exploitation.
- **Greedy Evaluation:**
  Greedy rewards are consistent (~452), indicating decent generalization.
- **Stability:**
  Intermittent reward dips even after convergence, hinting at learning instability.
- **Summary Insight:**
  Double Q-Learning improves overestimation bias but needs longer training to stabilize.

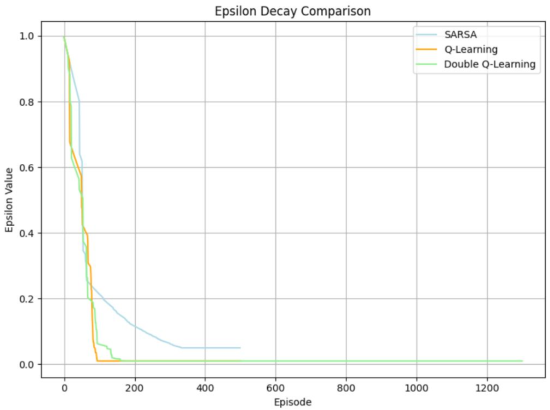




COMPARISON - Q-LEARNING VS SARSA VS DOUBLE Q-LEARNING

- **Reward Superiority:**
  Q-Learning exhibits the best average reward progression; SARSA is smoother but slower.
- **Exploration Uniformity:**
  All models begin with epsilon = 1.0 and decay similarly, ensuring fair exploration.
- **Greedy Ranking:**
  Greedy reward: Q-Learning ≈ Double Q > SARSA, with Q-Learning slightly ahead.
- **Learning Dynamics:**
  Double Q-Learning is more erratic, while SARSA maintains smooth learning; Q-Learning is faster.
- **Final Verdict:**
  Q-Learning strikes the best balance between speed, performance, and stability among the three.

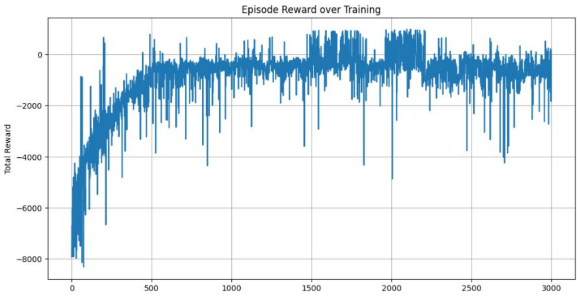


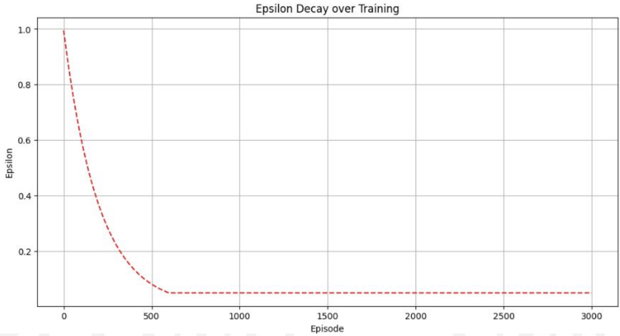


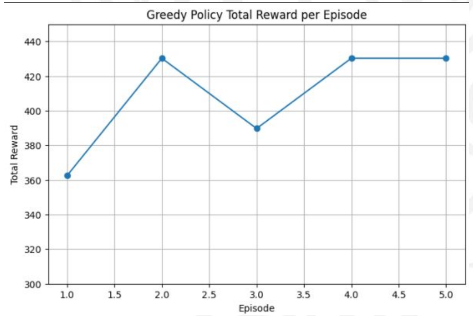
Using Deep RL Methods:

1. DQN

- Learn joint Q-function for 2 drones with centralized training → decentralized execution
- Architecture: state input →2 hidden layers → 36 joint actions
- Trained with experience replay, target network (update every 20 episodes), ε-greedy policy
- Achieved reward improvement from -8000 → ~0 after 2000 episodes; evaluation avg reward ~430

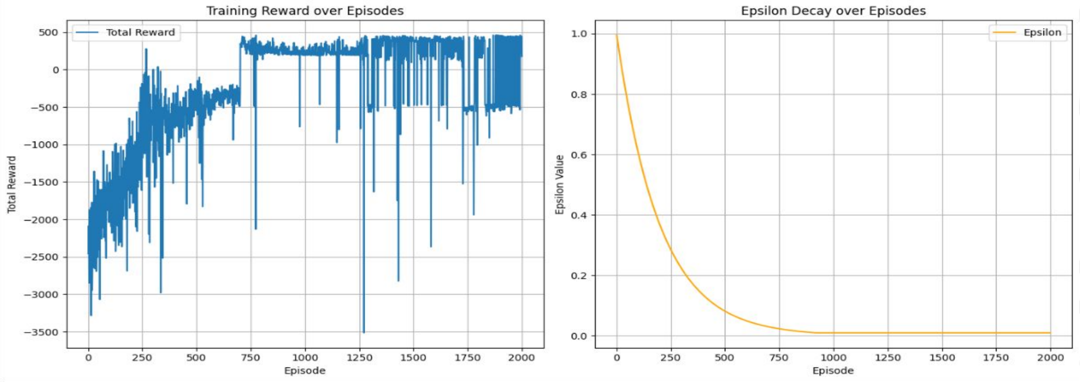


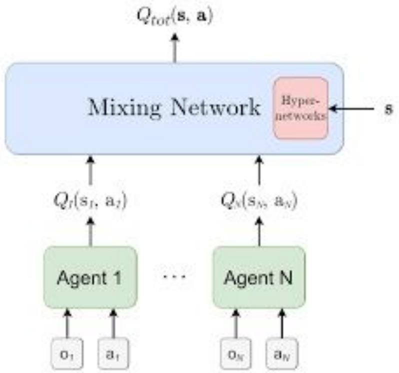


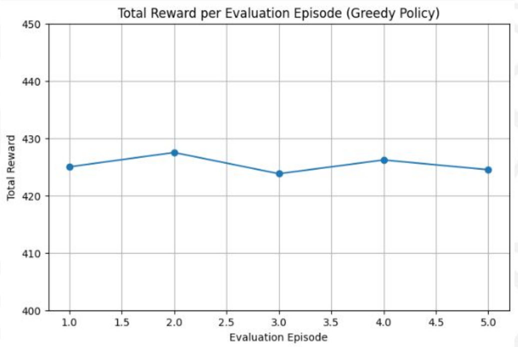


2. QMIX

- Implements centralized training → decentralized execution using a monotonic mixing network.
- Uses per-agent Q-Networks,Combines agent Q-values via a Mixer Network conditioned on global stateMixer.
- Experience replay buffer (size: 10,000) for learning stability
- Target networks for both agents and mixer updated periodically



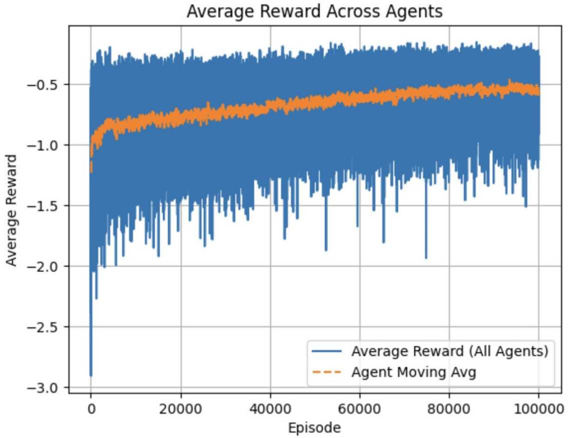


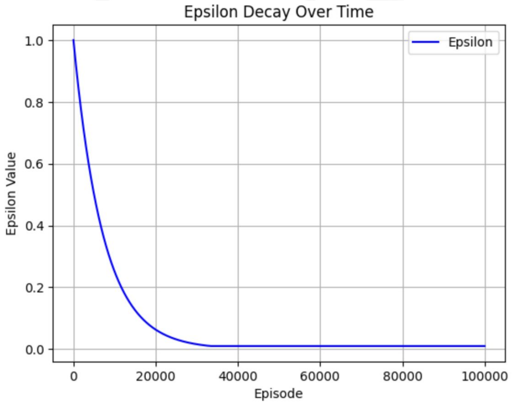


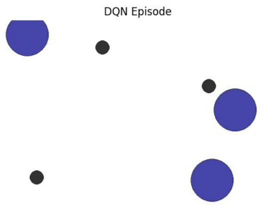
RESULTS- SIMPLE SPREAD V3

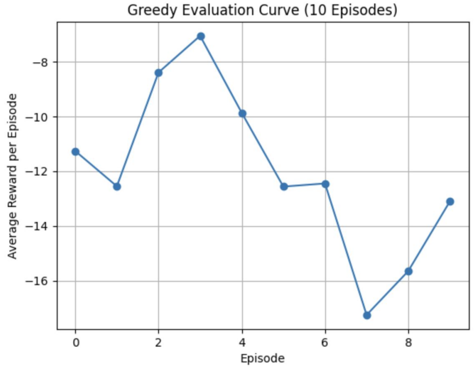
1. DQN

- **Reward Progression**: Average reward steadily improves but remains below -0.5.
- **Exploration Control**: Epsilon decays to ~0 within 20k episodes.
- **Stability**: Reward curve shows high variance across training.
- **Greedy Performance**: Greedy evaluation scores fluctuate between -8 and -16.
- **Overall Insight**: Indicates moderate learning with unstable test-time performance.



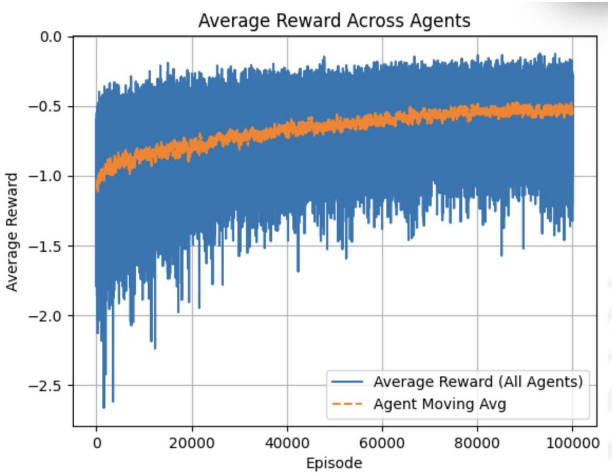


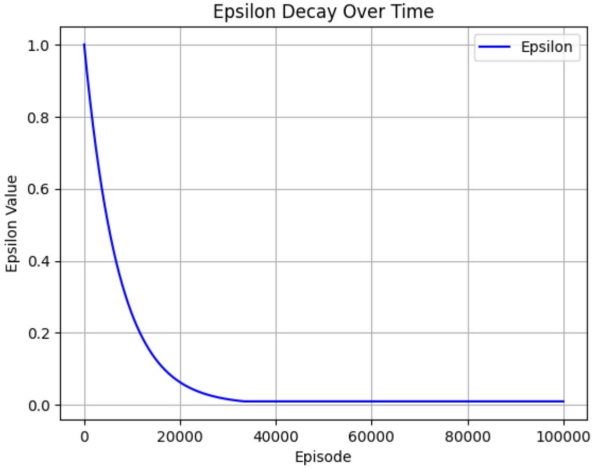


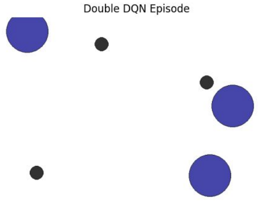


2. DOUBLE DQN

- **Reward Progression**: Average reward improves faster than DQN.
- **Exploration Control**: Epsilon decay behavior identical to DQN.
- **Stability**: Lower variance in training reward than vanilla DQN.
- **Greedy Performance**: Greedy evaluation reaches higher peaks (~ -8) than DQN.
- **Overall Insight**: More consistent policy learning observed during evaluation.

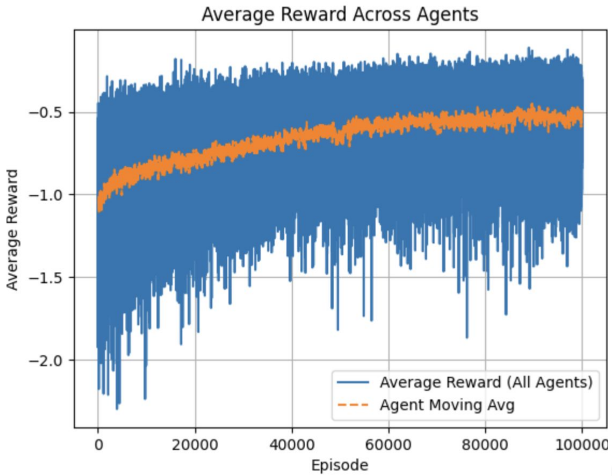


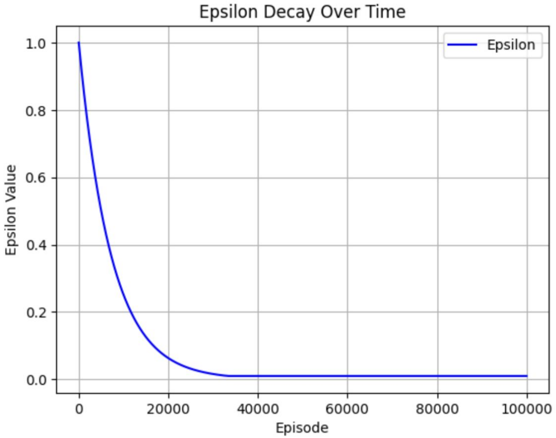


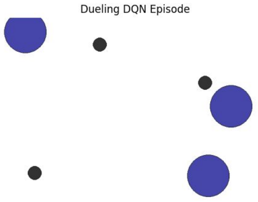


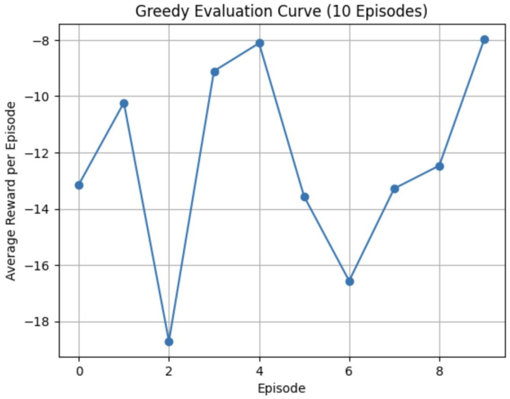
3. DUELING DQN

- **Reward Progression**: Highest average reward among all three models.
- **Stability**: Fast and stable convergence in training rewards.
- **Exploration Control**: Epsilon decay identical to others.
- **Greedy Performance**: Greedy evaluation rewards peak around -8 with minimal drops.
- **Overall Insight**: Best overall test-time performance and reward stability.



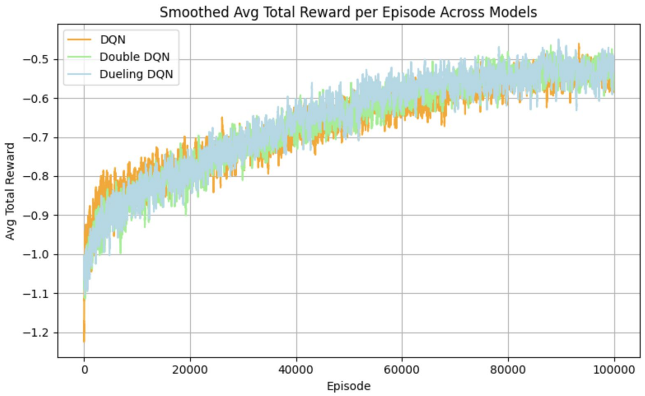


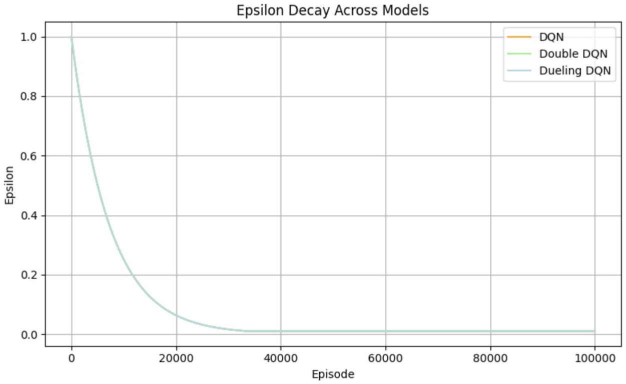


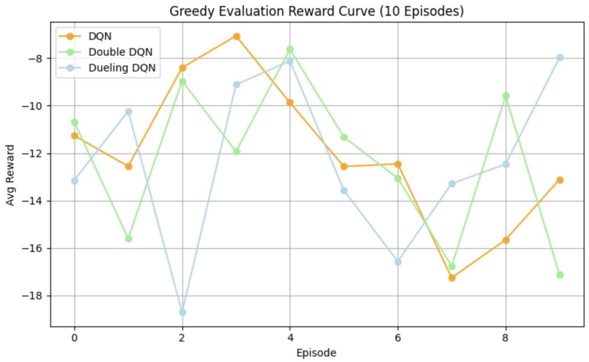


COMPARISON - DQN VS DOUBLE DQN VS DUELING DQN

- **Reward Superiority**: Dueling DQN leads in average reward curve.
- **Exploration Uniformity**: All models show identical epsilon decay trends.
- **Greedy Ranking**: Greedy evaluation: Dueling > Double > DQN.
- **Learning Dynamics**: Double DQN outperforms DQN in early and mid training.
- **Final Verdict**: Dueling DQN demonstrates the best consistency and generalization.







Key Observations & Summary

- Tabular RL methods (Q-Learning, SARSA, Double Q) effectively learned coordination in structured environments, with Double Q showing faster convergence and better reward stability.
- Deep RL algorithms (DQN, QMIX, Dueling DQN) scaled well to complex, high-dimensional settings and demonstrated superior generalization and consistency in both custom and benchmark tasks.
- Centralized Training with Decentralized Execution (CTDE), used in QMIX, proved effective in promoting cooperative strategies in multi-agent systems.
- The Drone Delivery environment highlighted real-world challenges like dynamic penalties, spatial constraints, and reward shaping under partial observability.
- Simple Spread v3 validated model performance in a standardized cooperative MARL benchmark, confirming the effectiveness of learned policies across agents.
- The project successfully met its objective of evaluating MARL algorithms in both custom and benchmark environments, demonstrating their effectiveness in learning coordinated, decentralized policies under partial observability.

Contribution

Team member	Aishwarya Virigineni	Nithya Kaandru	Prajesh Gupta Vizzapu
Project part	All team members contributed equally to environment design, reinforcement learning implementation, experimentation, visualization, and report preparation.		
Contribution	33.33%	33.33%	33.33%

Thank you