UNIVERSITY of TASMANIA
AUSTRALIA

Reinforcement Learning for Collaborative Decision Making

Research Presentation

Presentation by **Gemju Sherpa**

Masters of Information Technology and Systems | 2023

Student:
MITS, UTAS - Final Year

Degrees:
Bachelor of IT, Griffith University
Diploma of IT, Griffith College

Professional Goal:
Data Scientist/Software Engineer

Gemju Sherpa

Purpose

- Overview and highlight the key concept of my research
- Highlight research needs and key methods
- Show research findings
- Show limitation and future research directions

Presenter | Gemju Sherpa

Outlines

- Introduction
- Background
- Methodologies
- Experiments and Data Collection
- Key findings and Results
- Limitations
- Future Research directions
- Working simulation demo
- Q/A

Presenter | Gemju Sherpa

Introduction

Key Research Question

How to employ reinforcement learning for UAVs to learn effective collaboration in making decisions in a complex environment.

Research Domains

- Reinforcement Learning for Collaborative Decision Making
- Search and Rescue Tasks
- Autonomous Unmanned Arial Vehicles (UAV)
- Deep Reinforcement Learning
- Multi-Agent Systems domain

Objectives

Develop RL algorithm to train UAVs to perform collaborative decision making.

Analyze the learning performance of the agents in a given search and rescue environment.

Establish the baseline for MARL application search and rescue tasks.

Background

Why this research?

MOTIVATIONS

Recent advancements in AI and ML and its success stories.

Autonomous cars

Solving sequential decision-making in a complex environment with continuous action spaces

Games with human-level intelligence

Starting with the game of Backgammon (Tesauro and Keith, 1995), Atari games (Mnih, et.al. 2013) to the game of Alpha Go (Silver et.al 2016)

Robotics

Simple vacuum cleaner to Amazon delivery services

Resource Allocation

Minimizes the loss by efficiently allocating resources

Background

Our research Significance

No studies have ever been conducted in this problem domains → Expand the knowledge of AI/ML to collaborative decision-making with UAVs

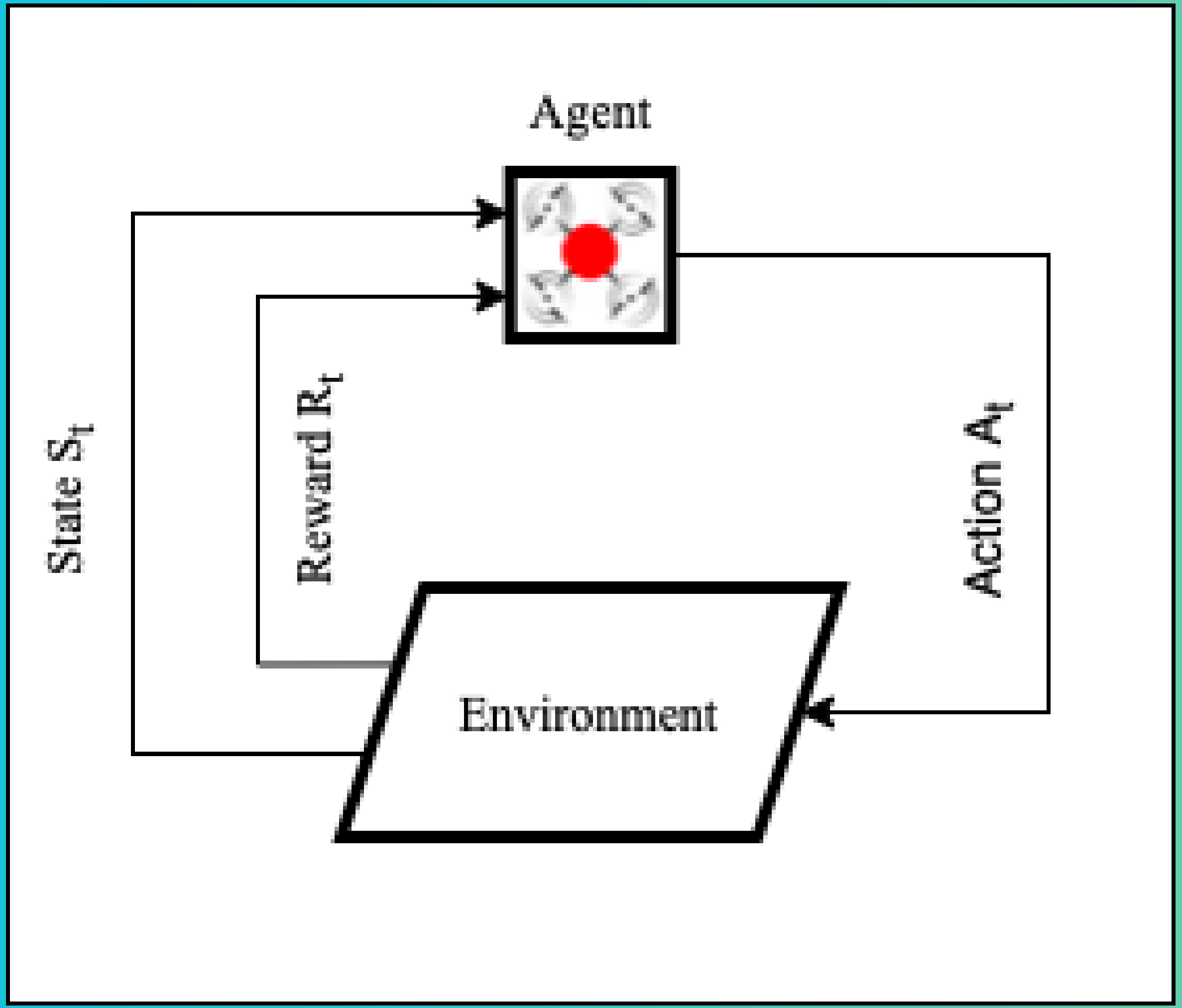
Provide insights into the capabilities of reinforcement learning for enabling UAVs to make effective decisions by working collaboratively. → Promising applications of UAV in Industry 4.0 and Defence.

Methodologies

Quantitative deductive approach

Methods

Reinforcement Learning



RL is an area of Machine Learning, where an agent performs an action in a given environment in order to maximize the reward.

Agent (a): Actor or any component that interacts within the environment

Environment (E): A problem space, where agent interacts and perform certain actions

Reward (R): A scalar numerical value, positive or negative, that is given based on the agents action choice

State (s): Part of the state, which the agent can observe

Methods

Reinforcement Learning

- In RL, the agents learn by doing. i.e. even if the agents are unknown in the environment, after a certain iteration, they will learn to make a move.
- The actions taken by the agents are greatly influenced by the reward it gets back after taking that action step.
- The agent will receive a positive reward for the correct step, while negative for incorrect step.
- The objective of the agent is to maximize the cumulative reward.

A value-function V, is the expected cumulative reward received by agent after completing the task successfully. This determines how good or bad the actions taken was. However, this does not give any information about next state.

Methods

Reinforcement Learning

Bellman optimality equation:

The value of a state is equal to the expected return for the best action from that state.

$$V^*(S) = max_a \sum_s T(s_t, a_t, s\prime)[R(s_t, a_t, s\prime) + \gamma . V^*(s\prime)]$$

V*(S) determines the action values for taking an action a at state s

The process of learning the optimal action values, also known as Q-Values, is called policy learning

a = argmaxQ (s, a)

Methods

Reinforcement Learning

Policy Update Methods

Monte Carlo Tree

Temporal Difference

Dynamic Programming

Deep neural network

Methods

Deep Q-Learning

Markov's Decision Process

$$Q(s,a) \leftarrow r + \gamma max_{a\prime} Q(s\prime, a\prime)$$

$$Q(s,a|\theta = Q(s,a|(\theta) + \alpha(r + \gamma max Q(s\prime, a\prime|\theta) - Q(s,a|\theta))$$

$$L(s,\ a|\theta_i) \ = \ (r \ + \ \gamma \ max Q(s\prime,\ a|\theta_i) \ - \ Q(s,\ a|\theta_i))^2$$

Deep Q-Learning
Where, Theta is the weight of the Q-Network

Methods

Online Policy Learning

Experience Replay

Every action a, state s, reward r, and next state s' of each iteration are stored in an experience buffer memory.

The policy Q will be updated by sampling random batch of samples from memory.

Methods

Algorithm pseudocode

Algorithm 1: Deep Q-Network with Experience Replay

Input: Initialize replay memory M with capacity N, exploration rate ϵ, discount factor γ, and neural network with weights θ

Output: Optimal Q-value function $Q(s, a)$

Initialize Q arbitrarily; 0 for all states, terminal states
for episode = 1, n **do**
    Initialize state s
    for step t = 1, t_n **do**
        if state s is not terminal, **do**
            With probability ϵ select a random action a, otherwise
            $a = \arg\max_a Q(s, a)$
            Execute action a
            observe reward r and next state s
            Store transition $(s, a, r, s\prime$ in M
            Sample a mini-batch of transitions $(s, a, r, s\prime)$ from M
            Update policy $Q(s, a)$
            $Q(s, a|\theta) \leftarrow Q(s, a|(\theta) + \alpha(r + \gamma maxQ(s\prime, a\prime|\theta) - Q(s, a|\theta))$
            Update the network weights by minimizing the loss function:
            $L(s,\ a|\theta_i) = (r + \gamma\ maxQ(s\prime,\ a|\theta_i) - Q(s,\ a|\theta_i))^2$
            Set $s = s_{t+1}$
    end for
end for

Table 3.1: Algorithm. Deep Q-Learning with experience replay buffer [12]

Methods

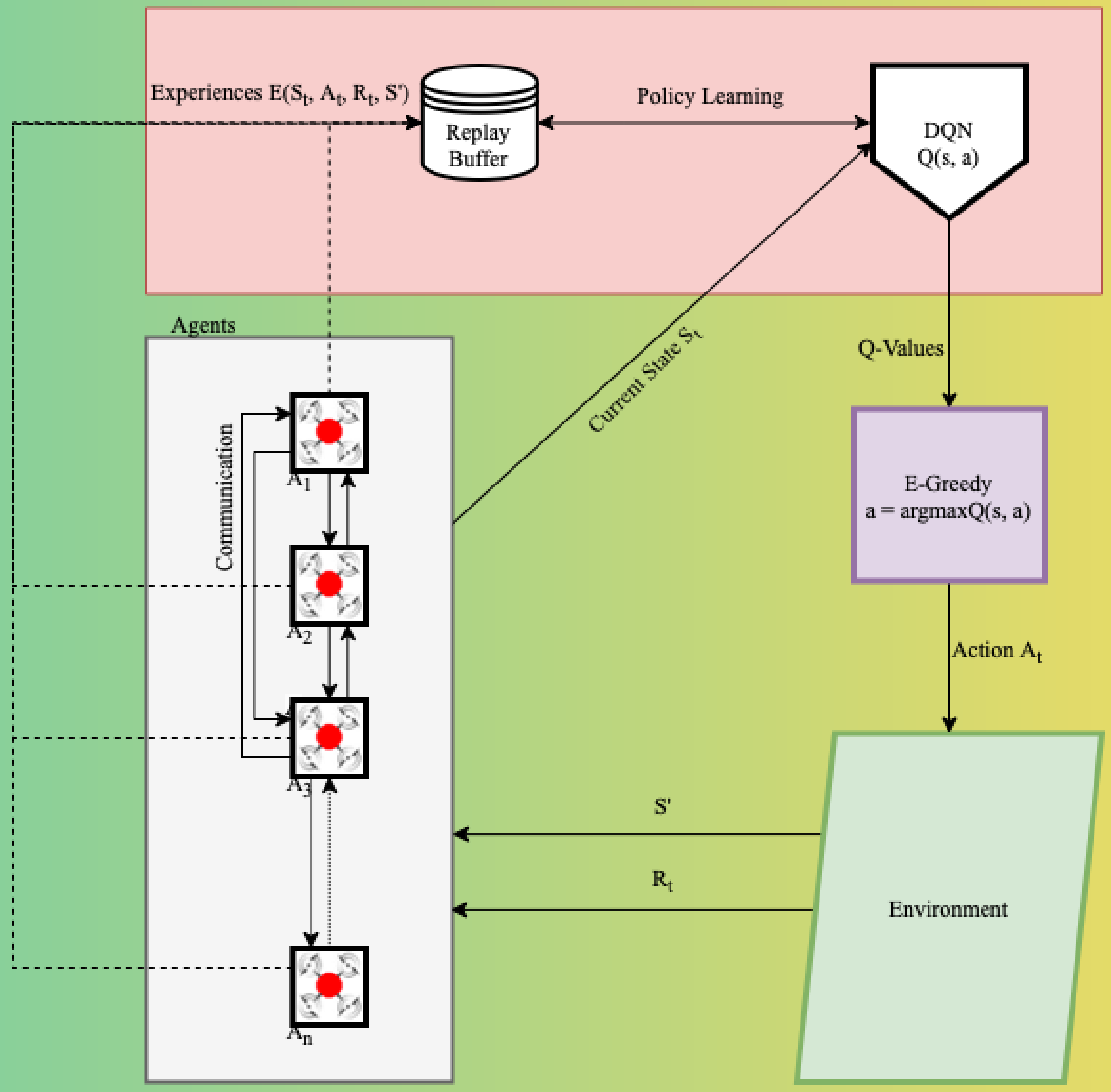
Multi-Agent Systems

Multiple agents interacting within an environment

- Can be cooperative
- can be competitive

Alpha Go

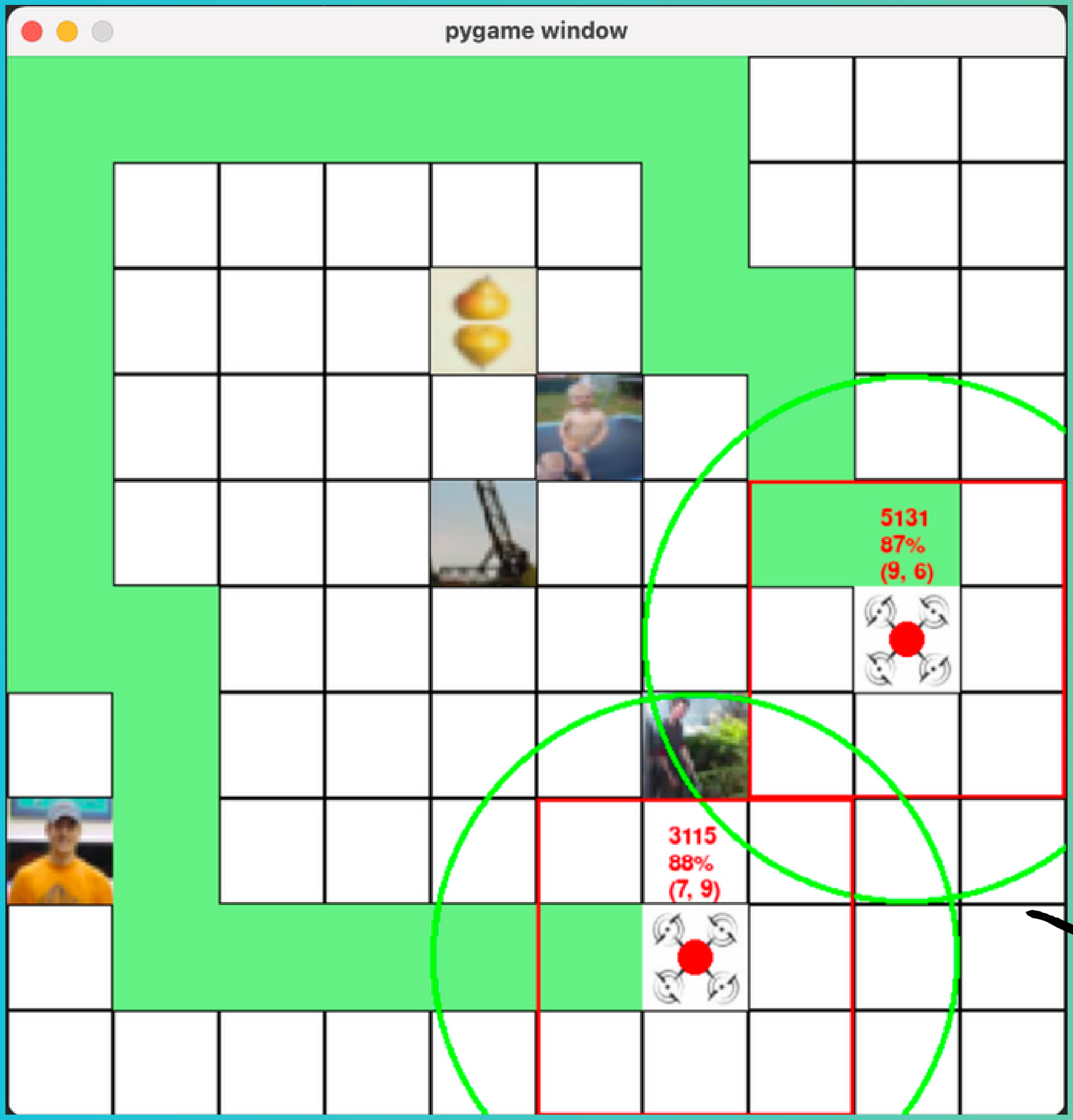
Lane changing System



Environmental Setup

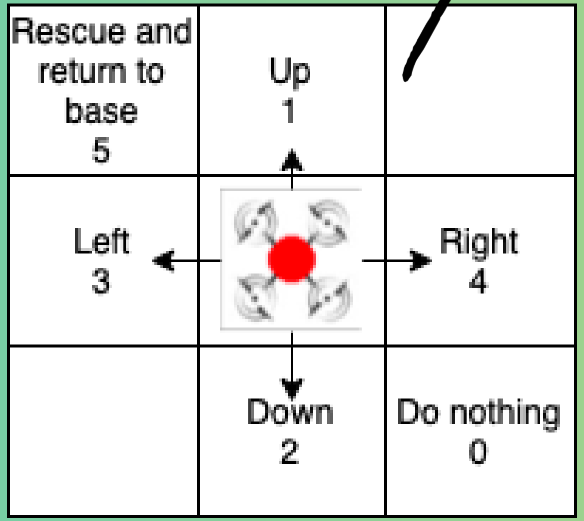
Setup

Environments



Overall environment with agents interacting

Action space



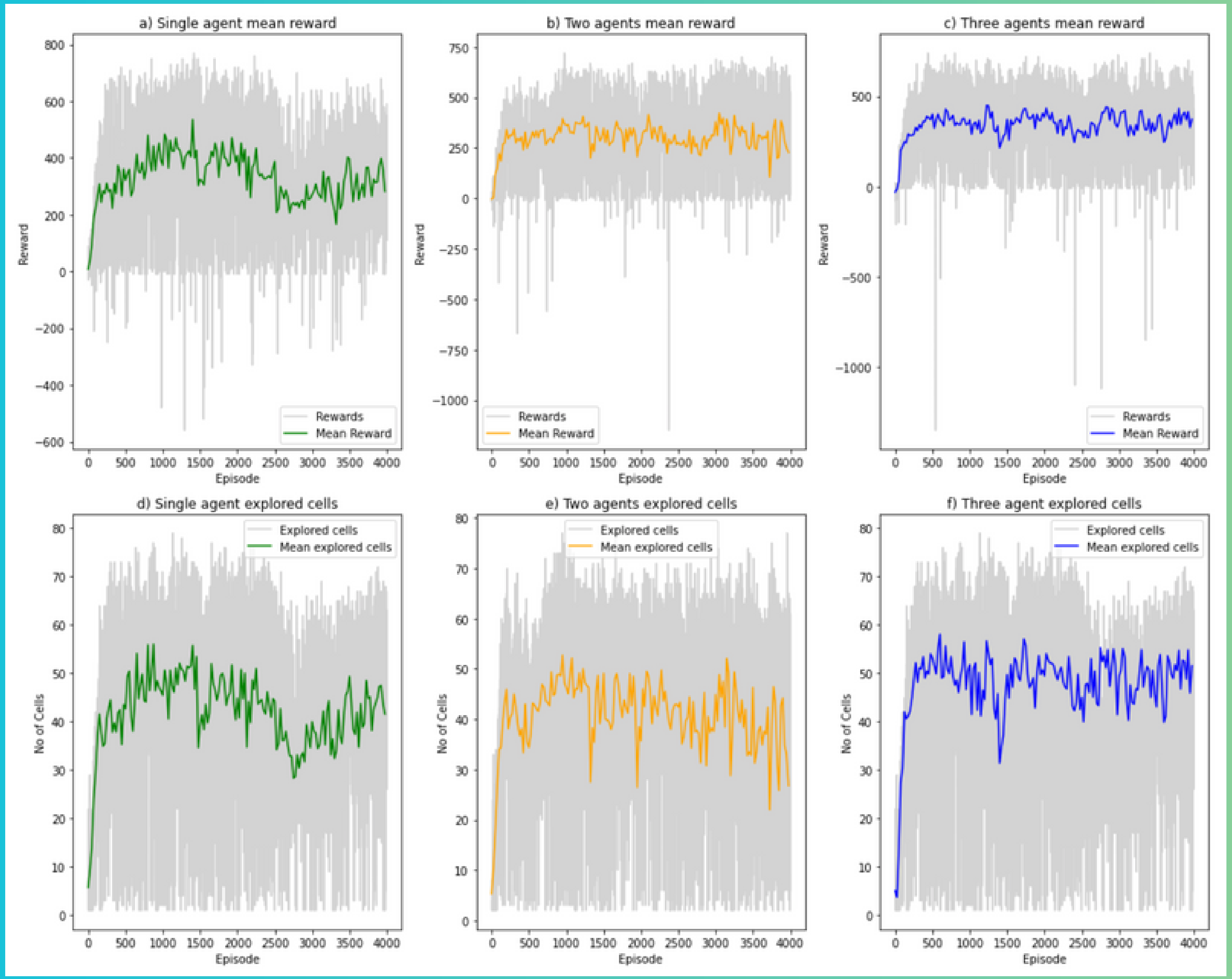
Reward assignment

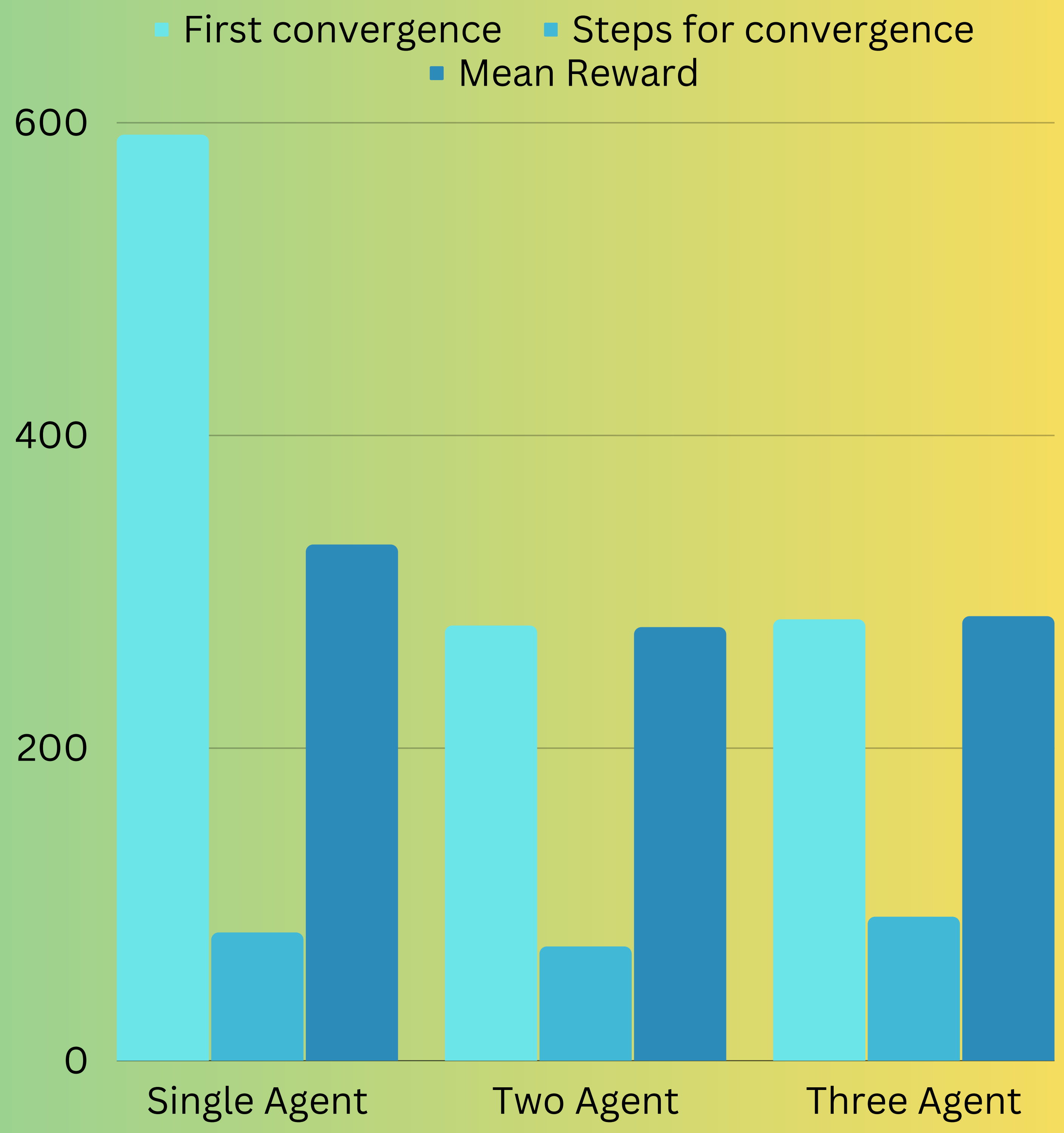
Action	Reward
Discovered new cell	10
moved to discovered cell	-10
Out of the board	-10
Did nothing	-10
Out of battery	-10
Rescued	100
Explored entire board	100

Results

Results

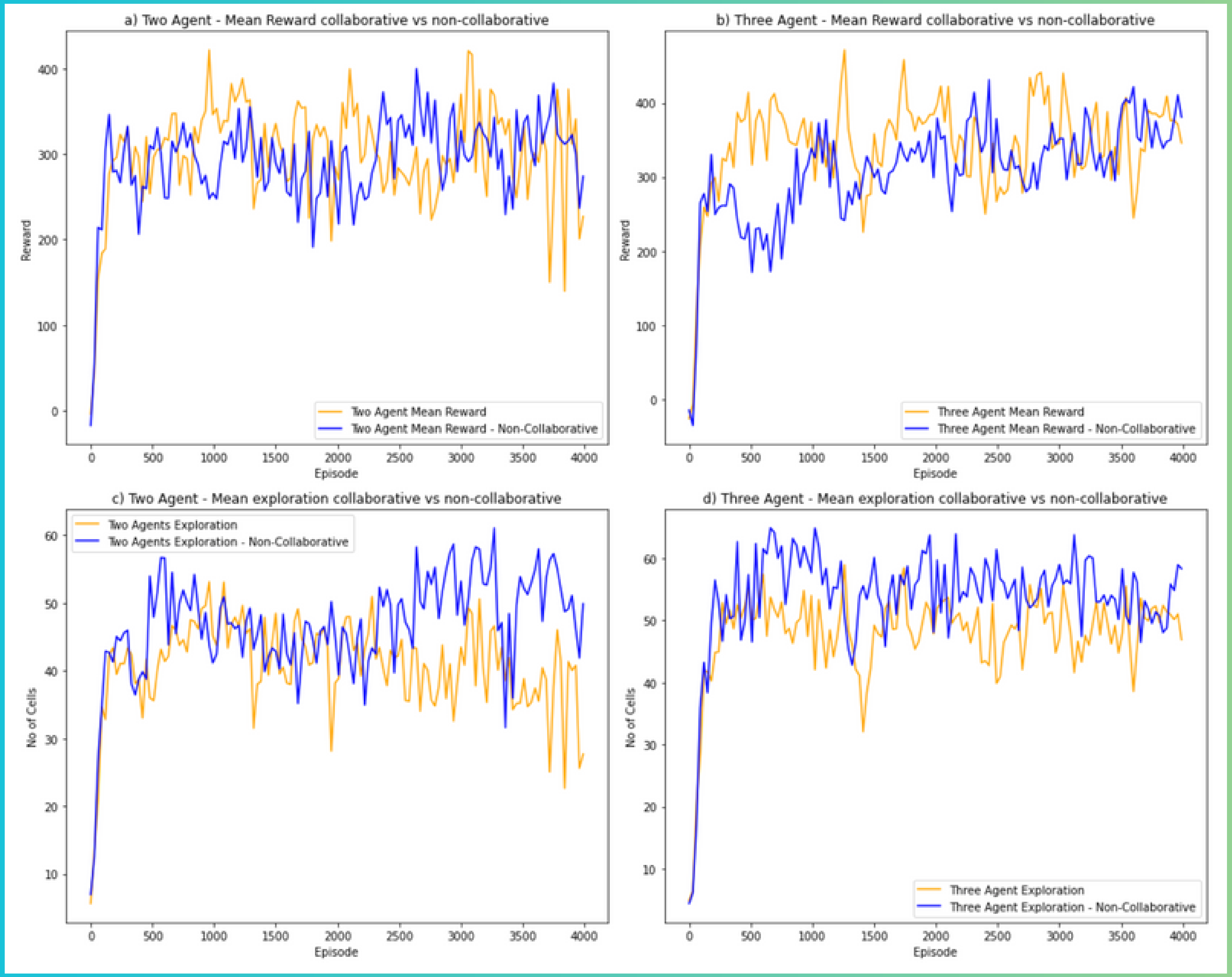
Collaborative

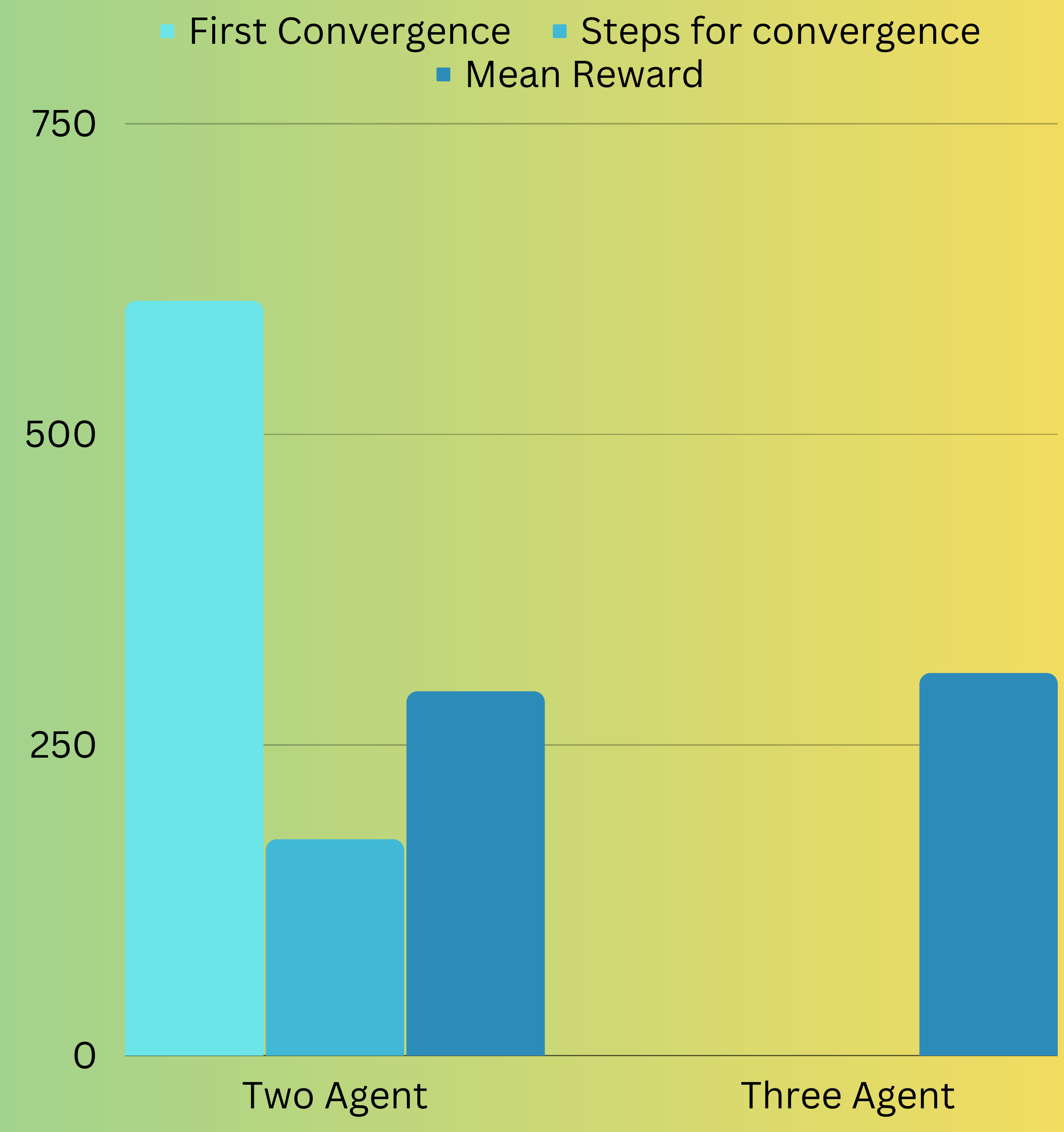




Results

Non-Collaborative vs Collaborative





Findings

Findings

- Our Deep Reinforcement Learning model successfully converged in 278 episodes of training with only 74 steps, when two agents are making decisions cooperatively
- Single-agent converged at 592 training episodes, whereas multi-agent learning converged at only 278 episodes, showing 50% faster and more effective
- Collaborative learning converged, while non-collaborative learning never converged

- Reinforcement learning techniques can be employed to train UAVs, where they can perform search and rescue tasks collaboratively in a complex real-world environment.

Limitations and Future Directions

Limitations

- Our conclusion and hypothesis validation is only based on training metrics
- Basic hyperparameter tuning
- Inadequate reward assignment function

Future Directions

- More validation of test results
- Train higher episodes, with the increased complexity of environmental variables
- Perform better and adequate parameter tuning
- Test the pretrained model in a real-world scenarios

Let's see in action


Thank you!

References

[1] By Gerald Tesauro and Tom Keith. Temporal difference learning and td-gammon, 1995.

[2] Silver, D., Huang, A., Maddison, C. et al. Mastering the game of Go with deep neural networks and tree search. Nature 529, 484–489 (2016).

[3] Playing atari with deep reinforcement learning. 12 2013.