Dissertation
KF7029
MSc Computer Science and Digital Technologies Project

Student Name: Mohanraj Muthumanickam

Student ID: W22056353

Supervisor Name: Lynne Connis

Second Supervisor Name: Tom Bartindale

MSc Programme: MSc Data Science

Dissertation Title: A Scalable Evaluation Framework for Intelligent Agents

Month and Year of Submission: August/2024

Academic Publication Venue: Expert System and Application Venue

Declaration

I declare the following:

(1) that the material contained in this dissertation is the result of my own work and that due acknowledgement has been given in the bibliography and references to **ALL** sources be they printed, electronic or personal.

(2) I have not use the services of any agency providing model or ghost-written work, nor have I employed unauthorised artificial intelligence in the preparation of this dissertation.

(3) the Word Count of this Dissertation is ....7835.....................

(4) that unless this dissertation has been confirmed as confidential, I agree to sections of the dissertation being placed on the eLearning Portal (Blackboard), if deemed appropriate, to allow future students the opportunity to see examples of past dissertations. I understand that if displayed on the eLearning Portal it would be made available for no longer than five years and that students would be able to print off copies or download.

(5) I agree to my dissertation being submitted to a plagiarism detection service, where it will be stored in a database and compared against work submitted from this or any other Department or from other institutions using the service.

In the event of the service detecting a high degree of similarity between content within the service this will be reported back to my supervisor and second supervisor, who may decide to undertake further investigation that may ultimately lead to disciplinary actions, should instances of plagiarism be detected.

(6) I have read the Northumbria University Policy Statement on Ethics in Research and I confirm that ethical issues have been considered, evaluated, and appropriately addressed in this research.

SIGNED: MohanrajMuthumanickam
DATE: 22/08/2024

Abstract

In this study, our aim to focus on the Multi-agent reinforcement learning approach or MARL in the competitive learning environment for shelf delivery tasks in the RWARE environment which is based on Gym environment structure. To learn agents in this complex and dynamic environment, we use two training techniques which are; Multi-Agent Deep Deterministic Policy Gradient (MADDPG) and DQN. In this regard, by comparing the performances of above-mentioned algorithms, one can study their capabilities in managing various competitive interactions and optimizing delivery tasks. Thus, our findings reveal the strengths and weaknesses of each algorithm and indicate a critical aspect of the opponent modeling and policy learning to build more efficient methods in multi-agent competitive environments. From this work, many research question germane to the design and application of efficient MARL algorithm for real-world environment such as warehouse automation have been answered.

Contents

1. Introduction

Reinforcement Learning (RL) [1] [31] is a subclass of machine learning where its learners (agents) must discover a sequence of activities within an environment that forms the best sequence of activities that yield maximum reward, optimum penalties inclusive. The core concept of RL is based on the idea of trying out different actions and learning from their results in order to reshape the policy according to the received experience. Indeed, this learning paradigm has been crucial in making it possible for agents to solve various problem domains on their own ranging from playing games all the way to controlling robots [2]. Initially, the RL was applied to the single-agent environments, and it involves an individual agent that learns within the environment to achieve a particular goal or perform some tasks. These single-agent systems provided the foundations for how an agent can acquire the best behaviors when working independently [3]. However, the drawback of single agent based system was gradually revealed as the researchers tried to extend the RL to more realistic and complex environment which is more towards the reality. This need hence gave rise to the Multi-Agent Systems (MAS) [4] in which many agents dynamically exist in one environment. Meanwhile, in full MAS, an agent not only has to learn how to achieve the maximum utility but also has to take into account other agents' actions and plans. Thus, this interaction creates new and more complex issues like coordination, communication or competition [5].

The transition from single-agent to the multi-agent system comprises the reasons like the enhancement of the entities' complexity, the requirement to work in teams, and competition in different areas [6]. For example, in robotics, several robots are required to fulfill particular tasks, for example, in robotic supply chain management, robotic organization of stores and warehouses, or in robotic soccer [7]. In the gaming industry, multi-agent systems are used for enhancement of the game environments by involving more than one player or character. In the same way, in economic markets, multi-agent systems are applied to understand the functioning of the market as a set of interacting individuals where the objective focuses on profit attainment. On this premise, competitive learning assumes some prominence. The source of competitive learning is based on situations where an agent cannot attain his/her objective without defeating rivals. This aspect of learning is helpful in those fields and professions with virtual close-to-life contestation processes like sports, games, and business. Like in many competitive games, agents (or the players) have to look for ways to get the upper hand and score past the opposition. The awareness of the opposite team's moves forms the basis of strategic games, which are designed to ensure that an individual triumphs. Market traders try to amass as much profit as possible in financial markets without any concern for the negative impacts of such acts on their counterparts [8]. Thus, competitive learning is an approach where researchers are specifically interested in algorithms and strategies agents would employ in competitive environments. They are more useful in areas where competition is key and RL is required to beat rivals to the game. There are issues of interaction dynamics between the competing learning algorithms and hence there is need to model opponents' behavior and plan for the next best action. Therefore, competitive learning is not just a way to create new application areas for RL and advance the state-of-the-art, but also to extend the horizons of conceptional development in strategic games, sports, and financial markets [9].

Multi-Agent Reinforcement Learning is formulated from the conventional reinforcement learning, where in this system; several agents learn and act in a given environment simultaneously. While in single-agent RL the reinforcing actions take place solely between an agent and the environment, in MARL the agents' actions can intrinsically affect each other or indirectly occur as a result of some other agents' actions [8].

Every fixer in a MARL system strives to gather the largest overall win in the fastest time as the agents can learn policies incorporating exploration and exploitation methods. The difficulty seen in MARL learning originates from the challenge that the agents experience in perceiving and in accommodating other agents in the environment – this greatly affects the system. In such environments the learning process is inherently more complicated because an agent has to design strategies that cover both cooperative and competitive scenarios and hence, provides a learning experience that is more sophisticated. Competitive learning is a specific aspect of MARL where the primary objective of the agents is to outperform their counterparts. Here they are not only learning to act in order to achieve their reward but also to reduce an opponent's reward; effectively turning the learning system into an environment with a zero-sum characteristic [10]. Competitive learning is most applicable in problems, which occurs in environments where the number of resources is scarce and they have to be gained in a contest. Competitive learning involves learning situations in which an agent's gain is an equal loss for another competitor; this may include; Sports competitions, strategy games, and financial markets. This competitive environment adds further levels of strategic complexity because agents have to both assess what their adversaries are likely to do and simultaneously devise responses to it in addition to carrying out their own planning. As for competitive learning algorithms the goal is to encourage agents to behave in adversarial manner and consequently come up with strategies that will be immune to strategies used by other competitors.

MARL can be characterized as a collection of multiple agents learning in an interconnected environment; this setting results in several difficulties compared to single-agent systems [11]. This is not the first problem where non-stationarity is a key issue as the environment constantly evolves with the other agents' actions and learning, which makes it hard for any one agent to discover constant policies. Another problem is the issue of scalability due to the fact that the interaction of agents and the size of the state-action space increases with the number of agents, this causes analytical and computational problems. The credit assignment problem bring in additional form of question, in that it becomes increasingly difficult to assign the resultant from the overall collaboration to a particular agent to determine whether that agent was right or wrong, hence fails to be useful for the agents to learn which action led to a particular success or failure [12]. Agent coordination and cooperation operationality are also essential but rather challenging, especially in situations where agents' operations involve coordinated and synchronized efforts or require settlements of harmonized objectives. Moreover, the task of managing the exploration-exploitation trade-off becomes more complicated as agents have to search for the optimal strategies and use known strategies to achieve the highest rewards, all in conditions of the uncertainty caused by other agents' behaviour. Last, it is stability and flexibility that are critical since the agents need to have mechanisms that would allow them to counter and respond to the future changes in other agents' strategy, especially since such a setup needs to be based on the learning models that are efficient and able to learn as well as adapt. That is why it is crucial to address these challenges in order to develop the MARL field further, and to effectively apply it in the realistic large-scale settings [13].

The communication between multiple agents in the environment is flexible in Multi-Agent Reinforcement Learning (MARL), and it is widely used in the actual world, including robotics [14]. In robotics, it allows for dealing with issues of cooperation and division of tasks among a group of robots, for example, to solve a difficult problem together, or be engaged in MARL as the robotic sports. In games and simulations, MARL is applied when entering competitive player versus player scenarios and when the aim is to improve the strategic decisions of the player by presenting realistic and challenging situations for the player.

It is also critical to understand and improve competitive learning in MARL to establish strong heuristics and cooperation strategies that would result in agents' better performance when operating against other competitors. A comprehensive understanding of competitive MARL, along with various aspects of designing, implementing, and assessing algorithms within the specified sectors such as robotics, gaming, and finance concepts, are also within the range of the study's concerns. Some of the limitations of the study may include the following; the nature of multiple interacting agents in an environment poses some constraints on MARL and challenges in increasing the size of MARL. We chose the MADDPG (Multi-Agent Deep Deterministic Policy Gradient) and DQN since the two are suitable for solving complicated multiple agent systems.

MADDPG works best for continuous action spaces and is able to learn polices with reasonable computational complexity in multi-agent environments where the actions of all the agents are dependent on each other. I also like the fact that it involves a decentralized critic with many agents that can learn from the actions of other permitted agents, which is important when dealing with competitive or cooperation MA environments. On the other hand, we select DQN which is a value-based algorithm since it is simple and performs very well in discrete action space. However, it is particularly strong in those cases where the action space is discrete and the state-action space can be thoroughly discerned with the help of the epsilon-greedy policy. Adding both of these algorithms, a broad spectrum of multi-agent environments can be covered while using the advantages of both, primarily MADDPG for continuous and high-dimensional spaces yet making use of the robustness DQN for discrete action cases.

2. Literature Review

Distributed Artificial Intelligence (DAI) which is a subfield of AI for about two decades encompasses systems with a number of autonomous entities that are present in a certain domain. This survey first defines Multiagent Systems (MAS) and provides diverse scenarios, issues, and techniques with the focus on machine learning approaches and then ROBOT SOCCER as a proper test bed while stressing the importance of non-robotic MAS to robotic MAS.[15]. The field of reinforcement learning is rapidly under development, especially after the application of deep learning applied to areas of control in robotics and games. However, most of the works remain limited to single-agent [3] context although many practical scenarios such as autonomous vehicular control, cooperative robotics [13], wireless networks, and stock exchange are inherently multiagent [16]. Each agent in these systems works on building a policy while acting with other entities in a common setting and therefore modifies their policy taking into account the others' actions [17]. Inspired by the recent breakthroughs in single-agent deep RL the MARL domain has gained popularity and has attracted a lot of interest in recent period where classic deep learning methods also substantially enriched the literature allowing the members of the community to get rid of the historical problems [18].

One can identify partially observable, competitive, multi-agent tasks that fall into this importance bucket. competitive problems do not have the difficulties in the evaluation which are connected with the general-sum game (for example, which opponents are evaluated against). the second noteworthy observation is that competitive problems have a clear correspondence to a large class of critical problems where a single user that manages a distributed system can state the general objective of the optimisation process [15] example, minimisation traffic or other inefficiencies. Many real-life situations require data that are either noisy or limited in some way. sensors, hence, coping with partial observability has to be done well. This good practice often involves restrictions. interactions which consequently generate the decentralised implementation of learned policies. However, there often entails the use of more information during training, which in most cases is conducted under restrained conditions or simulation. New developments on RL have been spurred by standardized evaluation, at the level of MARL benchmarks that push scalability of cooperative systems. Also, benchmarks for competitive MARL still did not find its suitable application and therefore the Fight Ladder, a real-time fighting game platform was revealed for further development of competitive MARL, as it offers a challenge in terms of dynamics, visual inputs, and the implementation of algorithms on the state of the art [19]. Self-play, where algorithms learn by playing against themselves, is has been considered vital in contemporary reinforcement learning (RL) for attaining superhuman performance notwithstanding the fact that, the RL theory is primarily developed and is mostly concerned with agents that compete against fixed environments with the effectiveness of self-play still open to question almost completely. This work focuses on self-play in competitive RL in a class of Markov games, presenting an algorithm referring to Value Iteration with Upper/Lower confidence Bound that has minimum regret and an explore-then-exploit algorithm that is sample-efficient and the first of its kind in competitive RL computation.[20]. In a recent work existing investigations of opponent modeling and intention inferencing struggle to provide clear descriptions and practical explanations of opponents' behaviors and intentions, limiting their applicability. This work introduces a novel approach using MADDPG to train agents and opponents in a competitive environment, employing TICC for behavior event extraction, and encoding this data into an opponent's behavior knowledge graph (OKG) to enhance policy learning through a question-answer system for querying historical information [21].

There are several other libraries (structured environment based on gym structure [22]) which are quite famous among researcher's and one of them is Exteted Python Educanth-Multi-Agent Reinforcement Learning EPyMARL framework is an improved version of the previous known as PyMARL [23], includes the choice of not sharing parameters, a feature that contrasts with the restriction put by the previous version which required parameter sharing only. There are certain aspects of EPyMARL, which ensure consistent procedures of the algorithms that is why the presented work is recommended to adopt as the framework for implementation depending on the algorithms basis. It clears the path for giving MARL a try on problems that would have been beyond anyone's imagination before, and allows for studying the agents' dynamics at a depth that was unimaginable until recently [24]. The algorithm we add in this model framework includes algorithms MADDPG [25], MAA2C [26],(IA2C [27], MAPPO [28] and IPPO [29]). Actor-critic approach (A2C) methods which also depends upon the increase of the reinforced agent observability as well as the reduction in compilation. complication for all the agents to display a better performance. Comparing it with other joint learning methods, the independent PPO where each of the agents has their own local value function fairs equally. The cooperative multi agent reinforcement learning to surpass well and at times perform better than the "centralized training with decentralized execution" strategy that can be deployed on the That is why such indicators as SMAC benchmark suite and the ability to catch up with jointly developed learning practically remain unaffected in the hyper-parameters estimation. MAA2C is a generalization of A2C (Advantage Actor-Critic) framework that is used for multi-agent scenarios. It allows the training of both individual and centralized policies using the same features and common value function optimization that enhances the working capability of the multiagents in complex situations and in making the right decision[30].

3. The Multi-Agent RWARE Environment

RWARE shown in Figure 1 is the Multi-Robot Warehouse environment that is a MARL simulation for modeling the warehousing situations where the robots are to transport and deliver shelves that have been ordered. Based on the principles of real-world automatization of warehouses, RWARE has several options, the extent of the warehouse, the number of agents, the type of rewards, etc. Agents act in a grid and can move and interact in a partially observable manner in order to make shelf delivery tasks efficient. The environment enables both the competitive and the cooperative setting which makes it well suited for the MARL algorithms.

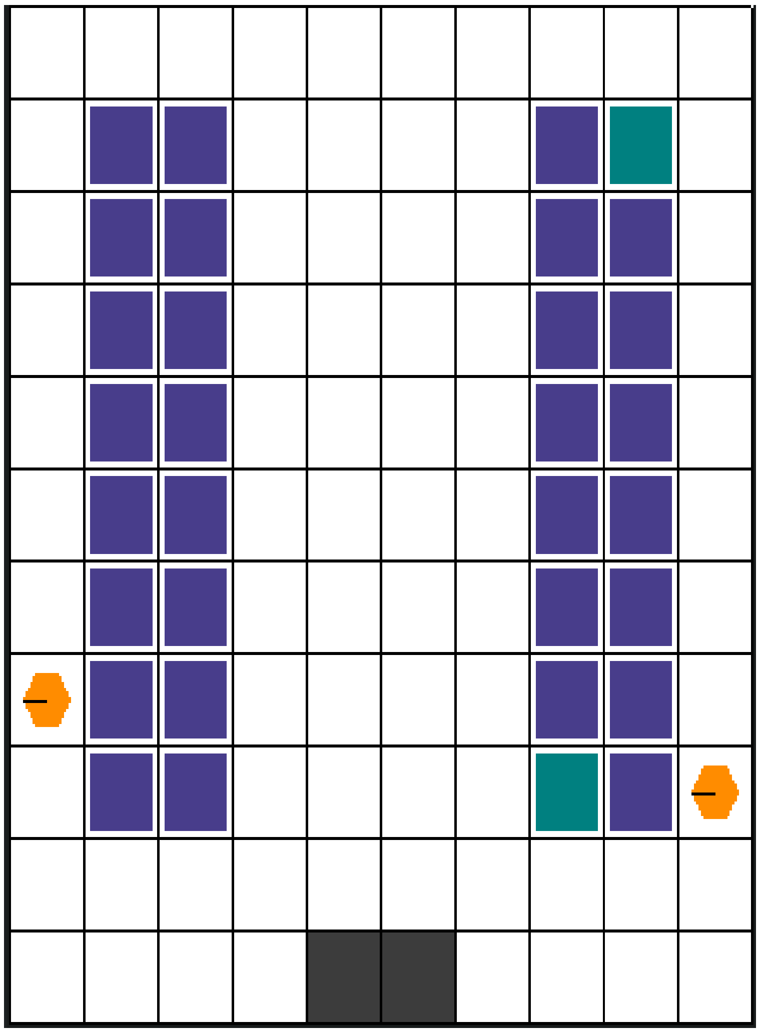

Figure 1: Rware environment

3.1 Action Space

The action space in the RWARE environment is also discrete, as shown in the above figure, where each agent possesses a limited number of available actions to be performed at a go. The available actions typically include:

- **Move Forward**: The agent attempts to move one step forward in its current direction.
- **Turn Left**: The agent rotates 90 degrees to the left.
- **Turn Right**: The agent rotates 90 degrees to the right.
- **Pick Up Shelf**: If the agent is adjacent to a shelf, it can pick up the shelf, provided it is not already carrying one.
- **Put Down Shelf**: The agent can place down a carried shelf if it is on a designated drop-off point.

These actions enable the agents to move about within a real grid-based warehouse environment, manipulate shelves, and perform delivery goals.

3.2 Observation Space

Each agent in the environment has only a restricted view of its environment, which makes the observation space for each agent partially observable. Each agent's observation is typically represented as a local grid centered around the agent, capturing:

- **Local Grid Information**: The presence of obstacles, shelves, and other agents within the agent's vicinity.
- **Shelf Status**: Whether a shelf is being carried or is stationary.
- **Agent's Orientation**: The direction the agent is currently facing (e.g., North, East, South, West).

This partial observability makes it that agents have to act under uncertainty, which amplifies the difficulty of the task.

3.3 Reward Space

Its peculiar reward space should motivate the agents into performing duties likely to improve chances of shelf delivery. The reward structure typically includes:

- **Task Completion Reward**: A positive reward is given when an agent successfully delivers a shelf to the correct location.
- **Movement Efficiency**: Agents may receive minor penalties for unnecessary movements, encouraging efficient navigation.
- **Collision Penalty**: Agents are penalized for colliding with obstacles, shelves, or other agents, promoting careful navigation.
- **Time-based Penalty**: A negative reward is sometimes applied for taking too long to complete tasks, motivating timely task completion.

The reward function can be made to be varied in order to encourage a certain behavior in the agents, for instance, cooperation between the agents or competition in completing tasks than the other agents in the system.

3.3.1 Dynamic Collisions and Environment Parameters

There are two types of dynamic collision in the RWARE environment – first, when agents, shelves or obstacles have collisions and interfere with such activities as delivering shelves and walking. These are penalized in the reward signal, meaning the agents are learning to move more tactfully, avoiding obstacles. The specified environment is fully adjustable and offers a range of configurable parameters that comprise size of the warehouse, number of agents and shelf configurations; these parameters either make the environment less or more complex. These parameters can be tuned and such flexibility allows the analysis of certain aspects of the real world allowing the researchers to test the capability of the multi-agent reinforcement learning in changeable and uncertain environments.

3.3.2 Configurability and Customization

One of the main characteristics of the RWARE environment is its high degree of configurability, which is essential for executing controlled experiments in reinforcement learning and multi-agent systems. Key configurable parameters include:

- **Size of the Warehouse**: On the environment settings, it is possible to change the warehouse size and due to this, the environment's complexity. The larger the warehouse the more difficult activities are because there are many areas to cover and many aisles to traverse.
- **Number of Agents**: Users can specify the number of robotic agents operating within the environment. More agents mean more complexity, coordination and collision will be a problem hence an important aspect in automating complex environment.
- **Communication Capabilities**: One of the features of the simulation is the ability of the researcher to switch on or off the ability of the agents to communicate with one another, and then view the effects of this decision in terms of how the agents perform the tasks and how effective they are in their teamwork.
- **Reward Settings**: RWARE also used the cooperative and competitive rewards format option only. The scheme of evaluation also can be cooperative and implies that each agent's work will be assessed based on the results of the common activity, which also speaks about the importance of teamwork. On the other hand, competitive environments pay agents based on performance hence encouraging competition and self-mobilization.

Dynamic collisions in the agents and in the RWARE environment are more specific than stationary, and they pertain to collisions involving physical contact between the agents or robots or, more broadly, the agents and the objects such as shelves or other obstacles in the grid-based warehouse. These collisions may interfere with the tasks, for example by blocking paths or by hindering shelf delivery and are usually punished in the reward system so that the player would avoid these behaviours and plan carefully. This environment has several factors, and some of them are the dimension of the grid, number of agents, shelf positions, and the difficulty of task. With these parameters, variations of the difficulty level are possible, from minimal objects and a corresponding small number of interactions to numerous and densely populated obstacles.

The fact that these parameters can be adjusted makes it possible for the researchers to introduce different scenarios that resemble real-life conditions of the warehouse, making such outcomes of the multi-agent reinforcement learning problem extremely beneficial for testing the flexibility and effectiveness of the multi-agent reinforcement learning algorithms in complex and unpredictable environments.

4. Methodology

4.1 MADDPG   Algorithm

The MADDPG is an extension of the well-known Deep Deterministic Policy Gradient (DDPG) that has been designed for use with multiple agents. specifies the DDPG (Deep Deterministic Policy Gradient) algorithm to multi-agent environment. It being particularly developed to cope with the problem of non-stationarity in environments. rather self-referential systems that may contain many agents, where agents play the role of the other as well as learn from each other. In MADDPG, every one of the agents possesses its unique policy which is learned by the actor' network and value function which is learned by the critic network jointly learning the actions of all other agents in the environment The notion of agency as defined here is still on an individualistic basis, not quite addressing the question how such agents are embedded in a social context, how they interact with other agents in the environment. This subsection goes further and explains the workings of the MADDPG. algorithm which involves; initialization, action, memory update and training process and target. network updates

4.1.1 Initialization

The MADDPG algorithm starts with the creation of the actor and critic networks for every actor. These networks are parameterized by weights and we initialize them at arbitrary in the start of the learning. Also, the target networks identical to each actor and critic network are set up. It also makes use of these target networks so as to improve stability of learning by using target values during the updating of the critic network. The target network are set to be the copy of the actor and the critic network with a time delay. The algorithm also speaks of a shared replay buffer which serves to store the experiences retrieved by every agent during the training phase.

4.1.2 Action Selection

In each contact with the environment, each agent chooses an action relative to its current policy. The action that is to be taken is derived in turn by applying its actor network on the observed state of the agent. The actor network produces a deterministic action which is further divided by the action bound so that the final action lies with in the required and allowed range of action. The selected actions are then performed in the environment and consequent state transitions, observed or calculated rewards, as well as the boolean terminal flags specifying whether the episode is done are stored in the replay buffer as an experience tuple.

4.1.3 Memory Storage and Replay Buffer

This is Memory Storage and Replay buffer Each training round also produces a memory storage and a replay buffer that contain the agent interactions with the environment by storing the state transitions so that the agent can utilize these states for referencing during the training process.

The replay buffer in the framework of MADDPG is of much importance as it contains the experiences of the agents. Experience tuple includes current state, all actions done by all agents, rewards for each agent, new state and is it the final state of an episode. The replay buffer is an important component of the algorithm since it enables the disruption of temporal relations inherent in the temporal sequence of the experiences and sample random sets of experiences during the training phase. Experience replay is a technique which makes the learning more stable and efficient if used in this manner.

4.1.4 Training Process

The training process in MADDPG is divided into two main updates: there are two updates; that of the critic and that of the actor. First of all, the critic network is updated for each of the agents. The critic network receives as a stimulus, the concatenated states and actions of all the agents in the environment. It computes and approximates the Q-value which stands for the cumulative expected reward of the agent at a state and a joint action. The target Q-value is determined through the mean of evaluating the target critic network and the target actions of all the agents from the target actor networks. The critic network is then updated using MSE that has been calculated by multiplying the difference between estimated Q-value and target Q-value by itself. Subsequent to updating the critic network, the actor network for every agent is updated. The actor update is done by optimizing the expected Q-value; this is done by minimizing the negative Q value that is estimated by the critic network. The gradients then that are used for the actor gradients are computed using the chain rule and back through the critic network. These gradients are then used to update the actor network in course of updating the policy.

4.1.5 Target Network Update

To make training more stable MADDPG used what is known as soft update to the target networks. Every training step, the weights of the target actor and critic networks are then trained with the mixed weights between the current networks and the target networks. In particular, through the weights of the target networks, the current weights joined with the weights of the target network previously used and regulated by τ . This allows making the target networks change slowly, which is beneficial for the targets during the critic update, this is a soft update.

4.1.6 Algorithm Execution

The steps included in the overall execution of the MADDPG algorithm are: Training the algorithm through a process involving interaction with the environment, update of the experience in the replay buffer and the update in the actor and critic network. This continues till the training comes to a stop or till a fixed number of episodes has been reached. The factors being assessed in the end are the learned actor and critic neural networks for each of the agents that can be used for real time deployment of the agents into the environment.

4.2 DQN Algorithm

4.2.1 Initialization

- **Q-Network**: A neural network is initialized to approximate the Q-values for each action given a state. The network takes the current state as input and outputs a Q-value for each

Algorithm 1 MADDPG Algorithm

```
1: Initialize:
2:     Initialize Actor and Critic networks for each agent
3:     Initialize target networks for each Actor and Critic
4:     Initialize Replay Buffer
5: for each episode do
6:     Reset environment and get initial state
7:     for each timestep in the episode do
8:         Select actions using Actor networks
9:         Execute actions, observe next state, reward, and done flag
10:        Store transition in Replay Buffer
11:        if Replay Buffer has enough samples then
12:            Sample a batch from Replay Buffer
13:            Update Critic networks by minimizing loss
14:            Update Actor networks using policy gradients
15:            Soft update target networks
16:        end if
17:    end for
18: end for
19: Functions:
20:     Actor: Dense layers followed by Tanh activation for action output
21:     Critic: Dense layers followed by linear output for Q-value
22:     ReplayBuffer: Add and sample experiences
23:     Soft update: Gradually update target network parameters
```

possible action.

- **Target Q-Network**: A copy of the Q-network is created and used as the target network. This network provides stable Q-value targets during training and is updated periodically or via a soft update.

4.2.2 Hyperparameters

- **Learning Rate (α)**: Controls the step size at each iteration during the optimization process.
- **Discount Factor (γ)**: Balances the importance of future rewards versus immediate rewards.
- **Exploration Rate (ϵ)**: Determines the likelihood of choosing a random action (exploration) versus the best-known action (exploitation). Initially set to 1.0 for full exploration.
- **Exploration Decay (ϵ decay)**: Gradually reduces the exploration rate after each episode, shifting focus toward exploitation.
- **Minimum Exploration Rate (ϵ_{min})**: The lower bound for ϵ, ensuring that some level of exploration continues throughout training.

4.2.3 Training Loop

The algorithm runs for a fixed number of episodes, where each episode involves interacting with the environment until a terminal state is reached.

- **Environment Initialization**: At the beginning of each episode, the environment is reset to a starting state. The agent observes this initial state.
- **Time Step Loop:**
  - **Select Action**: The agent selects an action using an ϵ-greedy policy. With probability ϵ, a random action is selected (exploration); otherwise, the action with the highest Q-value is chosen (exploitation).
  - **Execute Action**: The selected action is executed in the environment, resulting in a new state, a reward, and a done flag indicating whether the episode has ended.
  - **Store Experience**: The transition (state, action, reward, next state, done) is stored for later training.
  - **Train Q-Network**: The Q-network is updated by minimizing the loss between the predicted Q-value and the target Q-value, which is computed using the target Q-network.
  - **Update State**: The agent updates its current state to the new state obtained after taking the action.
  - **Episode Termination**: If the done flag is true, the episode ends, and the loop breaks.

4.2.4 Target Network Update

After each episode, the target Q-network is updated to slowly track the Q-network. This update can be done using a soft update, ensuring that the target network evolves smoothly and provides stable targets during training.

4.2.5 Exploration Rate Decay

The exploration rate ϵ is decayed after each episode to reduce the amount of random exploration. As the agent learns more about the environment, it shifts its focus from exploration to exploitation.

5. Simulation Setup

The simulation environment is RWARE, a comparison of two recent and very advanced reinforcement learning algorithms is performed, DQN and MADDPG. The RWARE environment, which is a competitive setting for several agents, each of which corresponds to a robot, to work together to perform tasks like picking up and delivering shelves to specified goal locations. MADDPG is a variant of the DDPG algorithm designed for multi-agent learning. MADDPG has an actor and critic network for each agent that are updated taking into account what other agents are doing, in order to learn policies that take into account the interactions in the environment. In this simulator both algorithms are assessed based on their learnability of good policies in the competitive RWARE setting. The performance of each algorithm is evaluated by the total rewards earned by the agents averaged over many episodes. The comparison is intended to illustrate the relative merits and

Algorithm 2 DQN Algorithm

```
1:  Initialize:
2:      Initialize Q-Network with random weights
3:      Initialize Target Q-Network with the same weights as Q-Network
4:      Set hyperparameters: learning rate $\alpha$, discount factor $\gamma$, exploration rate $\epsilon$, decay rate $\epsilon_{\text{decay}}$,
    minimum exploration rate $\epsilon_{\min}$
5:      Initialize Replay Buffer
6:  for each episode do
7:      Reset environment and get initial state
8:      for each timestep in the episode do
9:          Select action $a_t$ using an $\epsilon$-greedy policy based on Q-Network
10:         Execute action $a_t$, observe next state $s_{t+1}$, reward $r_t$, and done flag $d_t$
11:         Store transition $(s_t, a_t, r_t, s_{t+1}, d_t)$ in Replay Buffer
12:         if Replay Buffer has enough samples then
13:             Sample a batch of transitions from Replay Buffer
14:             Compute the target Q-value using the Target Q-Network:
15:                 $y_t = r_t + \gamma \cdot \max_{a'} Q_{\text{target}}(s_{t+1}, a') \cdot (1 - d_t)$
16:             Compute the Q-value from the Q-Network:
17:                 $Q(s_t, a_t) = Q_{\text{network}}(s_t, a_t)$
18:             Compute the loss:
19:                 Loss = $(y_t - Q(s_t, a_t))^2$
20:             Update the Q-Network by minimizing the loss
21:         end if
22:         Update state $s_t \leftarrow s_{t+1}$
23:         Decrease $\epsilon$ according to the decay rate
24:         Ensure $\epsilon \geq \epsilon_{\min}$
25:         if done flag $d_t$ is True then
26:             End the episode
27:         end if
28:     end for
29:     Periodically update the Target Q-Network with Q-Network weights
30: end for
31: Functions:
32:     Q-Network: Neural network with fully connected layers to predict Q-values
33:     ReplayBuffer: Stores transitions and samples them for training
34:     Soft update: Gradual update of Target Q-Network with Q-Network weights
```

demerits of each algorithm in dealing with competitive interactions, and to assess how quickly each can learn cooperative behavior in a common environment.

The number of episodes are high to provide the agents enough time to explore the environment and learn the optimal policy. In this work, we perform a comparative analysis of two state-ofic research directions of Multi-Agent Reinforcement Learning, namely, DQN and MADDPG, in the RWARE environment This environment in particular is meant to closely replicate a complex, multi-agent warehouse situation in which agents are robots that must collaborate in a common space to retrieve and deliver shelves to particular goal locations. Competition is baked into the RWARE environment, which is why it is a natural platform for assessing the performance of these algorithms in multi-agent domains.

Competition is baked into the RWARE environment, which is why it is a natural platform for assessing the performance of these algorithms in multi-agent domains.

In order to maintain consistency and to make a fair comparison between the DQN and the MADDPG algorithms all environment parameters are identical across the experiments. In particular, the warehouse is designed with 3 columns of shelves, where each column is 8 units high, and 1 row of shelves. Two agents are placed in this world, and each has a sensor range of 1 unit, so they can detect objects and other agents in their surrounded area. The agents are managed by a reward structure of RewardType. INDIVIDUAL, so that an agent is rewarded only for its own actions and not for the actions of a team.

Both algorithms are trained under the same environmental conditions with the following key parameters:

- **Number of Agents:** 2
- **Shelf Columns:** 3
- **Column Height:** 8
- **Shelf Rows:** 1
- **Sensor Range:** 1
- **Request Queue Size:** 2
- **Maximum Inactivity Steps:** None
- **Maximum Steps per Episode:** 500

Agents here take actions in an environment by taking a 'snapshot' of the current state, choosing an action according to some policy, and getting a reward based on how good that action was. The reward for each agent is recorded over all episodes of the simulator, and the cumulative reward is stored for each episode. 1000 episodes in total for the simulation, performance metrics (rewards) recorded and stored every 10 episodes to allow for continual monitoring of the learning process.

This same configuration on both DQN and MADDPG means that any comparison of the two can be made confidently about the true capability of each algorithm to manage competitive multi-agent interactions in the RWARE setting. The findings from these simulations help elucidate the relative advantages and disadvantages of each, especially as far as exploration strategy, policy-stability, and overall learning-efficiency of effective behaviors in complex, shared environments are concerned.

6. Results & Discussion

As shown in Figure 2 the MADDPG algorithm shows several important characteristics of the learning of the agents in the RWARE environment. The plots show the cumulative reward of each of the agents over episodes, which clearly depict how competitive the learning is. The smoothed reward plot of MADDPG is inclined upwards, according to the time, which implies that the agents are gradually improving their actions to achieve the highest reward. It also shows that the learning curve remains fairly constant with 'peaks and troughs' diminishing as training evolves. This stability is important in applications where agents have to learn in competing situations having to learn both the local environment and other competing agents. The performance patterns are almost identical in both the agents, although there are slight differences in their remunerations. This is in line with the fact that the MADDPG algorithm enables the two agents to learn adequate strategies that enable them to be competitive without one of them dominating the game. Minor technical changes in the value of rewards indicate that occasionally agents change their strategy but effectively they are redundant.

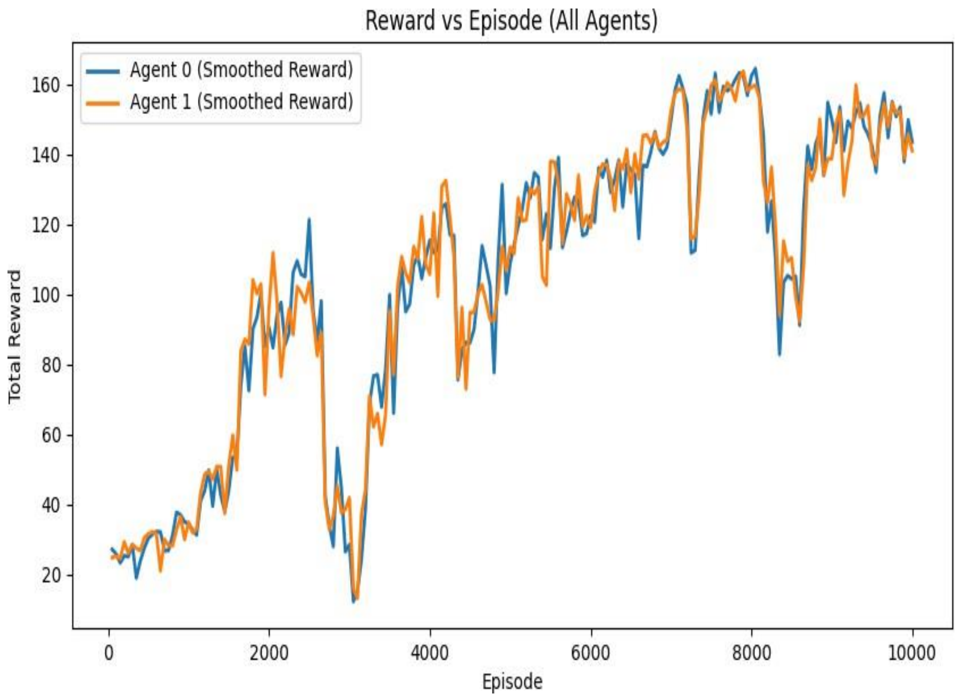

Figure 2: Smoothed Total Reward vs. Episode (All Agents) for MADDPG Algorithm

The first oscillations in the reward graph imply that the agents were, in fact, mapping the given context to find better ways of moving in it. During training, the agents turn to exploitation by choosing an action that has the highest reward among the still allowed ones. This behaviour is quite common in reinforcement learning and demonstrates the utility of the exploration exploitation trade off in the MADDPG architecture.

As shown in Figure 3, the DQN algorithm shows another learning process in comparison with

MADDPG. From the total reward plots of the DQN agents, one get a sense of the problems as well as opportunities of employing a more primitive form of learning like the value-based learning in a competitive multi-agent system. Comparing with MADDPG, there is a higher variance and less convergence in the DQN diagrams in terms of the rewards and performances. It is for this reason that it might be observed that DQN is not so effective in environments where the actions of other agents are likely to have a strong influence on the learning process, which would be characterized by a greater level of instability as compared to the level found in the present case. The agents themselves exhibit more variability; this is an evidence of their troubles in fine-tuning their strategies. Another factor that can be observed about the DQN agents is that their convergence rate is lower as compared to the MADDPG. The reward plot indicates that even with increasing the episodes; the agents do not perform as well as those trained under MADDPG. This slower learning process can be attributed to the absence of policy learning in DQN; while in some environments, strategic interactions are the key consideration.

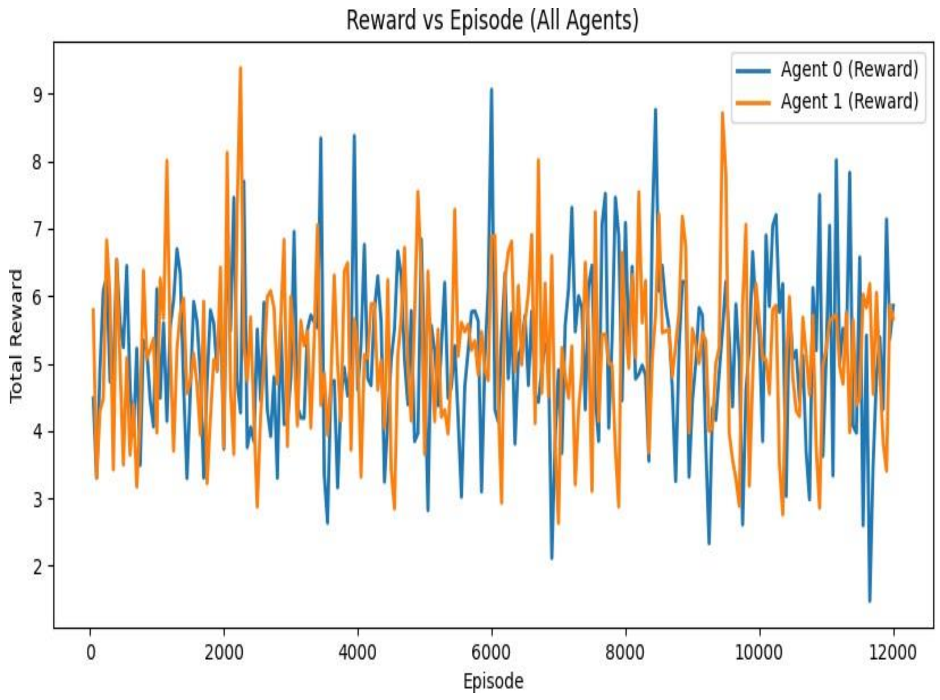

Figure 3: Total Reward vs. Episode (All Agents) for DQN Algorithm

From the Rewards graph it is seen that the DQN agents show a higher amount of aggression and exploration of the environment with frequent rise and drops in the rewards. But this aggressive exploration doesn't always in turn lead to superior performance as the agents are unable to converge to the best solutions. Such behavior imply that although exploration is crucial, DQN could need more sensitive adjustments of exploration parameters, when they are applied to competitive multi-agent scenarios.

6.1 Comparison of MADDPG and DQN in the RWARE Environment

Comparing MADDPG with DQN in light of the fact that it is a competitive reinforcement learning environment in the Robotic Warehouse environment provides understanding of the advantages and disadvantages of each. In this section, some distinctive features of their performance are described in detail, mainly learning stability, convergence, and adaptability, as well as scalability and computational effectiveness.

6.1.1 Learning Stability

MADDPG: Another major advantage of MADDPG is the stability of learning. The reward plots for MADDPG are smoother and there is an identifiable upward trend along with a tendency of the variability to reduce as the training progresses. This stability is due to that capability of MADDPG to learn a continuous policy for the agents; the agents can balance continuously and slightly revise their strategies in accordance with the feedback from the environment. The integration of deterministic policies and the addition of noise for improvisation assist the agents in gradually developing the set of rules, therefore the learning process in such environment is stable.

DQN: However, in learning stability, DQN is seen to have high variability more so than ANN. Indeed, the plot of the reward for DQN agents demonstrates the essential fluctuations and an absence of the constant growth, which testifies that the agents fail to sustain their learning continuously. DQN the action choice and the value function estimates are both discrete and this results in the change in policy of the agents as they switch between exploitation and exploration. This can make the learning process to be longer and irregular, especially when the environment changes, and the actions of other agents persistent affects the rewards.

6.1.2 Convergence and Performance

MADDPG: Overall, the process of convergence of MADDPG is not a very oscillating one and has a relatively fast convergence. The graphs representing the rewards achieved by the agents trained with MADDPG point to a progressive improvement of the performance of the agents. This enables the agents with the help of continuous action space as well as the actor-critic architecture of MADDPG to converge to a better policy space. This turns out to be especially beneficial when we are working in environments with multiple agents, where one has to learn how to counter strategies coming from other agents.

DQN: Whereas the convergence of DQN is slower and more fluctuating in comparison with MADDPG. These agents trained under DQN will take longer time to improve in performance and even the time taken to improve will not be so great a difference in the performance as that of the agents trained under other learning methods. This slower convergence is partly because of the DQN; this is a discrete method and it is not as flexible with the agents' moves in the environment. Second of all, DQN can be scaled by the 'curse of dimensionality' as the state-action space grows and learning can be extremely slow in large scale spaces such as RWARE.

6.1.3 Adaptability to Other Agents

MADDPG: Another of the key strengths of MADDPG is its flexibility in the multi-agent setting. By including actions of other learning agents in the learning process, MADDPG can adapt to the non-stationarity that would otherwise be posed by other learning agents. This is done by using the centralized training with decentralized execution method where each agent is trained to forecast the activities of the other agents in control and can perform better competition in the environment. Such flexibility is useful when new strategies of other agents in the RWARE environment have direct effect on the effectiveness of each such an agent.

DQN: The reason is that DQN is not as adaptable in the multi-agent environment because of the essence of its structure. However, traditional DQN assumes that the environment is static which is a major drawback in competitive multi-agent system. If the other agents are also learning and changing their behavior in the same process, then the environment cannot be stationary and for optimality DQN does not converge. This lack of adaptability generally leads to less than desirable performance, as observed in the reward plots where DQN agents are unable to more than occasionally outperform or maintain a pace with their opponents.

6.1.4 Computational Efficiency

MADDPG: Even though, MADDPG enhances learning stability, convergence, and adaptability during the training and show better results, it computational complexity is higher than DQN. The actor-critic architecture and the fact that essentially different policies must be learned for each of the agents only makes the task more demanding in terms of computations. Also, as in many PG methods, hyperparameters tuning is needed in MADDPG such as learning rates for the actor and the critic algorithms and so on, which contributes to the computational complexity. Still, the advantages of the MADDPG approach, specifically in such environments, usually compensate these computational expenses.

DQN: The main reason why DQN hardly takes any time to execute concerning computation than MADDPG is because DQN employs less value based learning than MADDPG. There is no need for policy gradients and discrete action spaces that decrease computational cost. However, DQN is quite simple, but it is not suitable for multi-agent environments due to its instability and inefficiency of learning. However, DQN is still a potential solution in possibly limited computationally environments such as RWARE, nevertheless the offered MAP scenario has to be relatively explained.

As observed from in Figures 4, for the total rewards collected throughout the learning process for both of the agents, the proposed MADDPG scheme gains a superior cumulative reward in contrast to the DQN. As mentioned previously, the figure of cumulative reward in the case of MADDPG climbs steeper than DQN and progresses more at a faster manner than the gradually built up curve of DQN. This trend indicates that MADDPG since it is able to do policy update that takes into consideration the actions of the other agents will provide better reward in this competitive multi-agent domain.

The learning rate (slope) of the reward for both the agents, Agent 1 in Figure 5 and Agent 2 in Figure 6 show differences when using the MADDPG as compared to the DQN algorithm. In both cases, we can observe a higher variance of the slope in MADDPG, meaning that learning in this algorithm is more active and probably unsustainable in terms of episodes. This is evidently expected to happen in a competitive multi agent environment where all agents are always in a constant learning mode of each other. Conversely, the learning rate of DQN is smoother, or let’s say, less volatile when observing the slope of a line, which indicates that this model learns slowly and steadily.

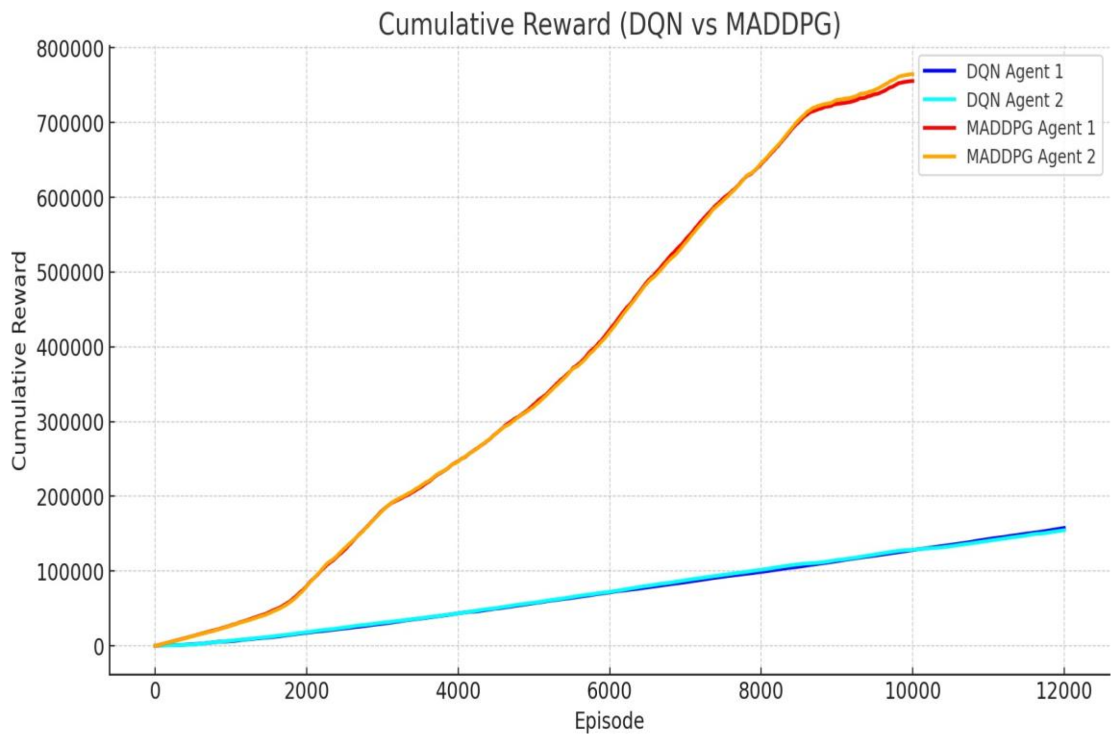

Figure 4: The total rewards accumulated over all episodes by both the algorithm.

In MADDPG a higher capacity to accumulate the rewards is shown but the variance and instability in learning is also generally higher. While the DQN is a much more stable algorithm, it may not be as swift in achieving the maximum of the rewards in such settings of the competition. Therefore, which of the two algorithms is used might depend on some considerations like need for stability to get some percent or massive returns.

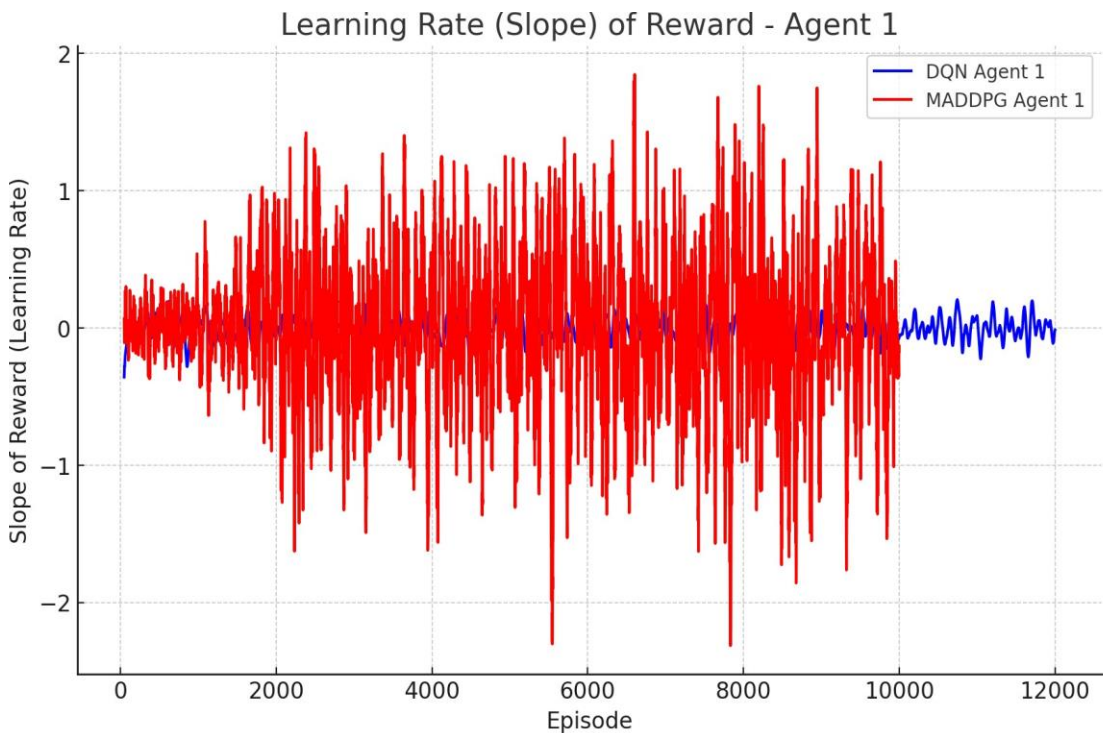

Figure 5: The rate of learning for each algorithm at different stages of training for agent 1

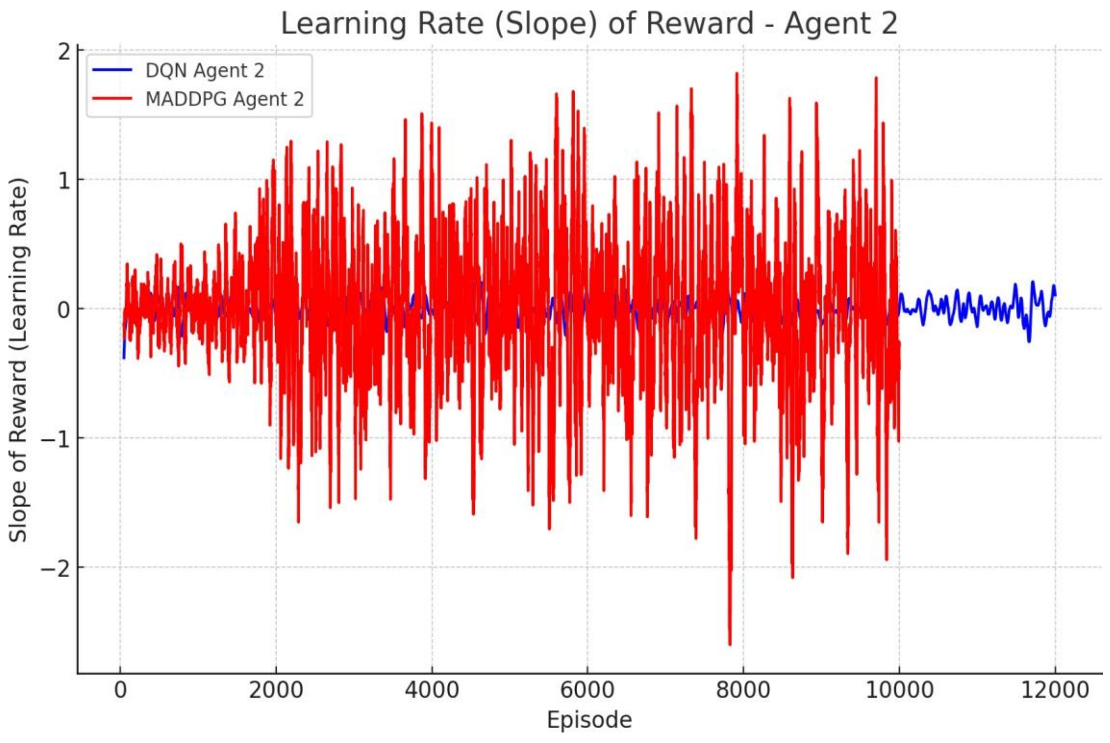

Figure 6: The rate of learning for each algorithm at different stages of training for agent 1

7. Conclusion

The analysis of the difference in the results of MADDPG and DQN in the RWARE environment allows to indicate the peculiarities of the algorithm’s work in the context of competitive multi-agent reinforcement learning.

MADDPG is a more stable approach for learning and converging as well as responsive with other agents aligning well with the complex environment. It is slower and more computationally expensive and it is sensitive to the choice of its hyperparameters.

On the other hand, there is **DQN** which is less complex and less computational than the first one but the learned policy is easily interpretable. However, its performance is constrained because it lacks the ability to deal properly with the non-stationarity and non-linearity of the applications in multi-agent environments, where learning is not as stable and converges more slowly than in simpler cases

All in all, as concerns tasks that grasp multiple interactions between agents, such as the shelf delivery in an automated warehouse, the MADDPG should offer greater performance. However, DQN can still be used especially where the computational complexity is not so high or where a number of computations are not so complex. The choice of algorithm is thus more of a question of specificity of the task, the available computing power and the need for interpretability versus accuracy.

Bibliography

[1] Barto, A. G. (1997) 'Reinforcement learning', *Neural systems for control*: Elsevier, pp. 7-30.

[2] Zhao, W., Queralta, J. P. and Westerlund, T. 'Sim-to-real transfer in deep reinforcement learning for robotics: a survey'. *2020 IEEE symposium series on computational intelligence (SSCI)*: IEEE, 737-744.

[3] Busoniu, L., De Schutter, B. and Babuska, R. 'Decentralized reinforcement learning control of a robotic manipulator'. *2006 9th International Conference on Control, Automation, Robotics and Vision*: IEEE, 1-6.

[4] Van der Hoek, W. and Wooldridge, M. (2008) 'Multi-agent systems', *Foundations of Artificial Intelligence,* 3, pp. 887-928.

[5] Perrusquía, A., Yu, W. and Li, X. 'Redundant robot control using multi agent reinforcement learning'. *2020 IEEE 16th international conference on automation science and engineering (CASE)*: IEEE, 1650-1655.

[6] Hao, J., Yang, T., Tang, H., Bai, C., Liu, J., Meng, Z., Liu, P. and Wang, Z. (2023) 'Exploration in deep reinforcement learning: From single-agent to multiagent domain', *IEEE Transactions on Neural Networks and Learning Systems*.

[7] El Hakim, A., Hindersah, H. and Rijanto, E. 'Application of reinforcement learning on self-tuning PID controller for soccer robot multi-agent system'. *2013 joint international conference on rural information & communication technology and electric-vehicle technology (rICT & ICeV-T)*: IEEE, 1-6.

[8] Zhou, W. J., Subagdja, B., Tan, A.-H. and Ong, D. W.-S. (2021) 'Hierarchical control of multi-agent reinforcement learning team in real-time strategy (RTS) games', *Expert Systems with Applications,* 186, pp. 115707.

[9] Riedmiller, M. and Gabel, T. 'On experiences in a complex and competitive gaming domain: Reinforcement learning meets robocup'. *2007 IEEE Symposium on Computational Intelligence and Games*: IEEE, 17-23.

[10] Fang, H., Zhang, M., He, S., Luan, X., Liu, F. and Ding, Z. (2022) 'Solving the zero-sum control problem for tidal turbine system: An online reinforcement learning approach', *IEEE Transactions on Cybernetics,* 53(12), pp. 7635-7647.

[11] Lee, H., Hong, J. and Jeong, J. (2022) 'MARL-based dual reward model on segmented actions for multiple mobile robots in automated warehouse environment', *Applied Sciences,* 12(9), pp. 4703.

[12] Yarahmadi, H., Shiri, M. E., Navidi, H., Sharifi, A. and Challenger, M. (2024) 'RevAP: A bankruptcy-based algorithm to solve the multi-agent credit assignment problem in task start threshold-based multi-agent systems', *Robotics and Autonomous Systems,* 174, pp. 104631.

[13] Zhu, X., Xu, J., Ge, J., Wang, Y. and Xie, Z. (2023) 'Multi-task multi-agent reinforcement learning for real-time scheduling of a dual-resource flexible job shop with robots', *Processes,* 11(1), pp. 267.

[14] Foerster, J., Assael, I. A., De Freitas, N. and Whiteson, S. (2016) 'Learning to communicate with deep multi-agent reinforcement learning', *Advances in neural information processing systems,* 29.

[15] Stone, P. and Veloso, M. (2000) 'Multiagent systems: A survey from a machine learning perspective', *Autonomous Robots,* 8, pp. 345-383.

[16]Lussange, J., Lazarevich, I., Bourgeois-Gironde, S., Palminteri, S. and Gutkin, B. (2021) 'Modelling stock markets by multi-agent reinforcement learning', *Computational Economics,* 57(1), pp. 113-147.

[17]Nguyen, T. T., Nguyen, N. D. and Nahavandi, S. (2020) 'Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications', *IEEE transactions on cybernetics,* 50(9), pp. 3826-3839.

[18] Hernandez-Leal, P., Kartal, B. and Taylor, M. E. (2019) 'A survey and critique of multiagent deep reinforcement learning', *Autonomous Agents and Multi-Agent Systems,* 33(6), pp. 750-797.

[19]Li, W., Ding, Z., Karten, S. and Jin, C. (2024) 'FightLadder: A Benchmark for Competitive Multi-Agent Reinforcement Learning', *arXiv preprint arXiv:2406.02081*.

[20] Bai, Y. and Jin, C. 'Provable self-play algorithms for competitive reinforcement learning'. *International conference on machine learning*: PMLR, 551-560.

[21]Ma, Y., Shen, M., Zhao, Y., Li, Z., Tong, X., Zhang, Q. and Wang, Z. (2021) 'Opponent portrait for multiagent reinforcement learning in competitive environment', *International Journal of Intelligent Systems,* 36(12), pp. 7461-7474.

[22] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J. and Zaremba, W. (2016) 'Openai gym', *arXiv preprint arXiv:1606.01540*.

[23] Samvelyan, M., Rashid, T., De Witt, C. S., Farquhar, G., Nardelli, N., Rudner, T. G., Hung, C.-M., Torr, P. H., Foerster, J. and Whiteson, S. (2019) 'The starcraft multi-agent challenge', *arXiv preprint arXiv:1902.04043*.

[24] Christianos, F., Papoudakis, G. and Albrecht, S. V. (2022) 'Pareto Actor-Critic for Equilibrium Selection in Multi-Agent Reinforcement Learning', *arXiv preprint arXiv:2209.14344*.

[25] Lowe, R., Wu, Y. I., Tamar, A., Harb, J., Pieter Abbeel, O. and Mordatch, I. (2017) 'Multi-agent actor-critic for mixed cooperative-competitive environments', *Advances in neural information processing systems,* 30.

[26] Iqbal, S. and Sha, F. 'Actor-attention-critic for multi-agent reinforcement learning'. *International conference on machine learning*: PMLR, 2961-2970.

[27] Chu, T., Wang, J., Codecà, L. and Li, Z. (2019) 'Multi-agent deep reinforcement learning for large-scale traffic signal control', *IEEE transactions on intelligent transportation systems,* 21(3), pp. 1086-1095.

[28] Lohse, O., Pütz, N. and Hörmann, K. 'Implementing an online scheduling approach for production with multi agent proximal policy optimization (MAPPO)'. *Advances in Production Management Systems. Artificial Intelligence for Sustainable and Resilient Production Systems: IFIP WG 5.7 International Conference, APMS 2021, Nantes, France, September 5–9, 2021, Proceedings, Part V*: Springer, 586-595.

[29] De Witt, C. S., Gupta, T., Makoviichuk, D., Makoviychuk, V., Torr, P. H., Sun, M. and Whiteson, S. (2020) 'Is independent learning all you need in the starcraft multi-agent challenge?', *arXiv preprint arXiv:2011.09533*.

[30] Ning, Z. and Xie, L. (2024) 'A survey on multi-agent reinforcement learning and its application', *Journal of Automation and Intelligence*.

[31]Papoudakis, G., Christianos, F., Schäfer, L. and Albrecht, S. V. (2020) 'Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks', *arXiv preprint arXiv:2006.07869*.

Appendix

- Here is the GitHub Repossitory link for RWARE, codes and plots: **https://github.com/Mohan-raj23/rware.git**

Original Feedback	**Changes made/Your Response to the Feedback**	**Where those changes can be found (Page numbers & section)**
The context and need for exploration in the space is clear, however the research objectives are non-specific, and are descriptions of activities to be conducted rather than articulating a research question to be answered. Some assumptions are also made about how much information the reader has about the teaching practices of digital forensics, making it difficult to parse the core challenges faced. The report does not describe the actual methodology under which data was collected, the study is neither reported or repeatable, making it hard to contextualise the data that is presented. Results are also mixed into this part of the report making it hard to distinguish method from interpretation. It is also not clear why a three armed study was created (360, 3D, 3d+headset), and so the technical implementation descriptions are missing context. The results are reported ad-hoc	I was asked to change the topic of my dissertation. Provided Clear abstract which addresses the methodology. The introduction is provided in detail about the work I will go through. Worked well on the literature review part for critical evaluation and identification of research gaps. Provided the practical work undertaken. Provided in detail results and analysis with actual results Provided detailed conclusion part and mentioned why one algorithm is better performing than the other.	Abstract section, Page number 1, 4, 10 Section 1, 2, 4 Page number 6, 13, Section 3 & 5 Page number 16, section 6 Page number 22, Section 7

with no structure in relation to the stated thematic analysis, and indeed it is unclear how much of the data was collected or what it represents. The use of graphs to represent this data demonstrates a misunderstanding of the type of analysis conducted. It is also not clear what the relationship is between the results presented and the discussion points raised. The presented discussion and findings do not answer the state research questions, or are not articulated as such. It is unclear how the presented findings are novel or offer a perspective outside the already identified literature.		