# MULTI-AGENT DEEP REINFORCEMENT LEARNING FOR COLLABORATIVE DECISION MAKING

by

Gemju Sherpa, MITS, BIT (TAS)

Submitted in fulfillment of the requirements
for the Degree of Masters in Information Technology and Systems

Department of Information and Communication Technology
University of Tasmania
May, 2023

UNIVERSITY of TASMANIA

I declare that this thesis contains no material which has been accepted for a degree or diploma by the University or any other institution, except by way of background information and duly acknowledged in the thesis and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due acknowledgment is made in the text of the thesis.

Signed: _____
Gemju Sherpa

26/05/2023

Date: _____

Signed: _____
Gemju Sherpa

26/05/2023

Date: _____

# ABSTRACT

This paper presents a study on the implementation and evaluation of a Multi-Agent Reinforcement Learning (MARL) algorithm based on Deep Q-Network (DQN) for Unmanned Aerial Vehicles (UAV) to make collaborative decisions to perform search and rescue tasks. The agent(s) - UAVs, were trained using centralized-learning-decentralized-execution methods in a simulated environment. The simulation board was represented as a $10 \times 10$ matrix, with objects and humans that are represented by positive integers, while known and unknown areas of the board are represented by -1 and -2, respectively. Each agent was initialized with specific attributes such as position, battery life, observation range, and object-carrying status. The DQN model, implemented using TensorFlow and Keras, consisted of a densely connected neural network with $128 \times 128$ input, hidden layers, and 6 output neurons. A replay buffer of size 50,000 was used for offline learning, and the training process involved updating the weights of the training network based on the target network.

The experimental results demonstrated that multi-agent learning outperformed single-agent learning, with collaborative decision-making being more effective than non-collaborative. The multi-agent system successfully explored the environment and achieved stable reward accumulation. Additionally, the study compared collaborative and non-collaborative learning scenarios and found that collaborative learning resulted in higher rewards, while non-collaborative learning showed better exploration rates. These findings validate the efficacy of the MARL algorithm for search and rescue tasks and highlight the benefits of collaborative decision-making in multi-agent systems.

**Keywords**: Reinforcement learning, multi-agent reinforcement learning, search and rescue, deep Q-network, collaborative decision-making, exploration.

# ACKNOWLEDGEMENTS

I would like to take this opportunity to express my deepest gratitude and appreciation to several individuals who have been instrumental in the completion of this project.

First and foremost, I would like to extend my heartfelt thanks to Dr. Son Tran for his exceptional guidance, unwavering support, and invaluable expertise throughout this research endeavor. Dr. Tran's profound knowledge and insightful feedback have been instrumental in shaping the direction of this work, and I am truly grateful for his mentorship.

I would also like to extend my sincere appreciation to Mr. Guan Huang for his significant contributions and support during the course of this project. His dedication, assistance, and collaborative spirit have been invaluable, and I am grateful for the valuable input and suggestions.

I would like to acknowledge the support and encouragement provided by my colleagues and friends who have been a source of inspiration and motivation throughout this journey. Their constructive discussions, encouragement, and friendship have made this research experience more fulfilling and enjoyable.

Furthermore, I am grateful to the academic community and researchers in the field for their valuable contributions, which have served as a foundation for this work. Their dedication to advancing knowledge and sharing insights has been an ongoing source of inspiration.

Lastly, I would like to express my deepest appreciation to my wife for her continuous support, understanding, and encouragement. Her love and belief in my abilities have been my constant driving force, and I am truly grateful for her presence in my life.

Without the collective efforts and contributions of these individuals, this project would not have been possible. I am sincerely thankful for their support, guidance, and encouragement, which have been integral to the successful completion of this endeavor.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# Chapter 1

# INTRODUCTION

Reinforcement Learning (RL) is one of the most widely studied and adapted Machine Learning techniques to solve the most complex sequential decision-making problems. It is used to train agent(s) to make decisions, where optimal policies will be improved gradually, by trial and error [11] based on a reward achieved by the agent(s); by taking a sequence of actions in a given environment. The most successful and popular adaptation of RL is the game of Go [4], autonomous cars [12], Atari games [2], robotics, etc.

Based on the problems aimed to solve and the complexity of the environment, there exists more than one agent, and modeling these agents have become a problem in reinforcement learning. Multi-Agent Reinforcement Learning (MARL) aims to address these challenges. MARL is a sub-field of Reinforcement Learning concerned with multiple agents in an environment, where agents take an action seeking to maximize total reward within the environment. The agents work towards a common goal by undertaking the action that guarantees maximum reward – sharing the knowledge accumulated during the exploitation and exploration of the environment. The MARL is one of the most popular techniques to solve sequential decision-making problems in a competitive and cooperative environment with multiple autonomous agents [28]. In the recent decade, MARL has been widely adopted in many industries and various problem spaces – including games, robotics, automobile, marine, and networking are some of the prime examples. However, despite the success of MARL in many industries, very little study has been conducted in the field of search and rescue environment and the application of reinforcement learning for collaborative decision-making to UAVs is often neglected. The potential of MARL with the integration of Deep Q-Learning to train agents (each UAV) to make decisions independently to perform search and rescue tasks is enormous. This research aims to unlock the potential and the application of MARL for autonomous UAVs for the search and rescue task through the study of multi-agent systems in complex environment settings using deep reinforcement learning. This study can be considered a novel contribution to the field of search and rescue.

The introduction chapter will explore research motivations, the significance of MARL, research gaps, and objectives of the research, and are structured as follows. The motivation section briefly outlines the motivation that led this study, followed by the significance of MARL. In this section, we will provide a description of the sig-

nificance of multi-agent systems. The research gaps section describes the need for this study, while the research objectives section provides a brief overview of research objectives followed by the outlines of the overall structures of the paper.

## 1.1 Motivation

The motivation for this research stems from the significant advancements in multi-agent deep reinforcement learning observed in recent years. These advancements have showcased the remarkable capabilities of multi-agent systems in a wide range of continuous and discrete environments, including the successful control of Atari games [2], the Alpha Go game [4], self-driving cars [12], and resource allocation tasks. These achievements have demonstrated that agents can learn to perform complex tasks with human-level accuracy, even in the face of challenges such as sparse and noisy reward variables, partial observability of the environment, and heterogeneity among agents [2, 17]. Furthermore, the application of multi-agent reinforcement learning (MARL) has been increasingly employed in autonomous cars for cooperative driving and lane-changing tasks, as evidenced by recent studies [29]. These developments highlight the tremendous potential of MARL and deep reinforcement learning techniques, prompting the exploration of their application in the context of search and rescue tasks performed by autonomous UAVs. By leveraging the power of MARL and integrating deep Q-learning, this research aims to unlock the vast potential and address the application gap of MARL in collaborative decision-making for autonomous UAVs operating in search and rescue environments.

## 1.2 Significance of MARL

The significance of this research lies in its exploration and application of Multi-Agent Reinforcement Learning (MARL) in the context of autonomous Unmanned Aerial Vehicles (UAVs) for search and rescue tasks. While MARL has gained significant attention and success in various domains, its potential and application in the search and rescue domain, particularly with UAVs, have been largely overlooked.

The integration of MARL with Deep Q-Learning presents a novel and promising approach to address the challenges encountered in search and rescue operations. By enabling individual UAV agents to make independent decisions and collaborate in a complex environment, this research aims to unlock the potential of MARL for enhancing the efficiency, effectiveness, and safety of search and rescue missions.

The practical significance of this research extends to multiple fronts. Firstly, it contributes to the field of search and rescue by introducing a new paradigm that leverages the power of MARL and deep reinforcement learning techniques. The successful application of MARL in search and rescue operations can revolutionize the way autonomous UAVs are deployed, enabling them to autonomously navigate and make informed decisions in dynamic and hazardous environments.

Secondly, this research has the potential to significantly enhance the capabilities of search and rescue teams, empowering them with intelligent UAV agents that

can operate collaboratively and adaptively. By efficiently exploring and exploiting the search space, these autonomous UAVs can contribute to timely and effective decision-making, ultimately leading to improved outcomes in critical life-saving situations. Moreover, the outcomes of this research have broader implications beyond the search and rescue domain. The insights gained from studying multi-agent systems in complex environments using deep reinforcement learning can be applied to other fields, such as disaster management, surveillance, and environmental monitoring. The research findings can inform the development of intelligent systems that can operate autonomously, cooperatively, and adaptively in a wide range of challenging scenarios.

Furthermore, this research addresses a notable gap in the existing literature, which has paid limited attention to the application of MARL for collaborative decision-making in the search and rescue domain. By shedding light on this neglected area, this study contributes to filling the research gap and expanding the frontiers of MARL.

## 1.3 Research Gaps

Despite the extensive study of Multi-Agent Reinforcement Learning (MARL) applications across various industries, the majority of these studies have focused on fully observable and discrete environments. While significant progress has been made in applying MARL to autonomous vehicles and robotics, there is still a notable gap when it comes to fully cooperative multi-agent systems operating in partially observable environments [19].

The current state of research in this area falls short of achieving the benchmark of fully cooperative multi-agent systems in highly complex environments. The challenge lies in effectively coordinating and leveraging the collective intelligence of agents in scenarios with partial observability, where agents have limited access to information about the environment and the actions of other agents. This limitation hampers the full realization of the potential of MARL in complex real-world settings.

Hence, there is a pressing need for further investigation and study of MARL in diverse environmental settings. By addressing the research gap in understanding and harnessing the capabilities of fully cooperative multi-agent systems in complex and partially observable environments, this research aims to bridge the existing knowledge gap and unlock the full potential of MARL. The findings from this study will contribute to advancing the field by developing novel algorithms, techniques, and strategies that enable effective collaboration and decision-making among agents in challenging environments.

By filling this research gap, this study seeks to push the boundaries of MARL and provide valuable insights into the design and deployment of autonomous systems that can effectively navigate, learn, and cooperate in complex and dynamic real-world settings. Furthermore, by addressing these research gaps, this research can pave the way for the development of more robust, adaptable, and efficient multi-agent systems in various domains, including search and rescue, robotics, and au-

tonomous transportation.

## 1.4 Research Goals

In this study, we utilize the pre-existing multi-agent deep reinforcement learning concept and extended it to our problem space with improved parameter settings within the complex real-world continuous and partial observable environment to train UAVs to learn to make decisions cooperatively. Although the goal is to study the full potential of MARL within complex environments, the training agents will be done within the discrete $10 \times 10$ matrix simulation with added complexity such as object detection, and search and rescue as shown in Fig 3.1. The high-level goal of the study is to design and experiment with the application of Deep RL in UAV by formulating the problem as MARL systems of complex environments for the real-world use case for search and rescue tasks.

## 1.5 Research Question

This project aims to investigate the novel ideas of reinforcement learning techniques for autonomous UAVs in a complex search and rescue environment and expand the existing findings in the field of multi-agent systems. The key research question of this research is 'How to employ reinforcement learning for UAVs to learn effective collaboration in making decisions in a complex environment'. On a broader level, key questions such as 'is RL techniques can be applied to search and rescue tasks?', 'is a multi-agent system better than a single agent for search and rescue tasks?', and 'Is collaborative decision-making effective for learning of UAVs?' are expected to be tested, and generalized conclusions based on the test results.

The algorithm developed for multi-agent systems using deep reinforcement learning for the research is expected to outperform single agent acting in the same environment. Therefore, the conclusion is expected to be drawn based on the performance of a single agent versus multiple agents making decisions collaboratively.

## 1.6 Structure of the Paper

The remaining sections of this paper are organized as follows:

Chapter 2: Literature Review
In this chapter, we provide a comprehensive review of previous work related to reinforcement learning. We discuss the foundations of reinforcement learning and delve into the research on multi-agent deep reinforcement learning, collaborative and competitive RL, and their applications. By examining existing literature, we establish a solid understanding of the current state of the field and identify the key advancements and challenges.

Chapter 3: Research Methodology
In Chapter 3, we present the research methodology employed in this study. This

includes a detailed description of our research philosophy, research design, and strategy. We outline the environment in which our experiments are conducted and present the methods and baselines utilized for comparison and evaluation. This chapter provides a clear framework for understanding the experimental setup and the methodologies employed in our research.

Chapter 4: Environment Setup and Experimental Results
In this chapter, we focus on the setup of our experimental environment and present the results of our experiments. We describe the specific configuration and implementation details of the environment, highlighting any necessary modifications or adaptations. Furthermore, we present the findings and outcomes of our experiments, supported by quantitative analysis. This chapter serves as a crucial section for understanding the empirical evidence and performance of our MARL model.

Chapter 5: Discussion, Limitations, and Future Directions
Chapter 5 entails a comprehensive discussion of the results obtained in Chapter 4. We analyze and interpret the experimental findings, considering their implications and significance within the context of our research objectives. Additionally, we address the limitations and potential constraints of our study, providing a critical evaluation of our methodology and results. Furthermore, we discuss possible avenues for future research and highlight areas that warrant further investigation.

Chapter 6: Conclusion
In the final chapter, we conclude our research by summarizing the key findings, contributions, and implications of our study. We revisit the research objectives and research questions, highlighting how our research addresses the identified research gaps. Additionally, we provide a concise overview of the main outcomes and insights gained throughout the research process.

# Chapter 2

# Literature Review

In this section, we will present some of the most recent advancements in the field of AI and Reinforcement Learning as comprehensive literature by comparing and contrasting ideas of astonishing work that has been conducted in this field by different professionals and individuals as a systematic literature review. The literature review has conducted in a mixed method of both chronological ordering (beginning to date) and thematic settings. Terminologies used in these sections are DQN referring to Deep Q-Network, MARL referring to Multi-Agent Deep Reinforcement Learning, A2C referring to Actor-Critic-Agent, and R referring to reward in all contexts.

## 2.1 Reinforcement Learning

Reinforcement learning has been studied greatly and has received huge traction and success over the last decades. The idea of reinforcement learning to develop an intelligent system to solve complex problems by trial-and-error methods was considered a revolution during the early evolution of artificial intelligence. Although reinforcement learning has shown great potential, there have been many paradigms that require further improvements and study. One core foundation of reinforcement learning is the value approximation function. In a relatively simple problem space, a linear function to estimate action value in a state V(s) using linear function approximation, ultimately converges local optimal [23]. However, the linear function value approximation is not efficient and hence the study of convolution neural networks has been introduced with reinforcement learning known as deep reinforcement learning.

The earliest success of reinforcement learning was the self-playing game algorithm known as TD-gammon to play the game of Backgammon; TD-gammon is a Temporal Difference based neural network that learns itself as an evaluation function by playing against itself [24]. Following the success of Backgammon, the same principle has been extended to many other games such as chess, Go, and checkers, however, it has not performed as in Backgammon which led scientists to explore other options [2].

The true power of reinforcement learning received significant attention soon after

the success of OpenAI Atari games in 2013, which showed online policy learning - Deep Q-Learning, that combines stochastic minibatch updates and experience replay for Q-learning as an efficient deep reinforcement learning algorithm in these games [2]. These games were implemented using deep Q-Network (DQN) as a function approximator, where Q-Network was trained by minimizing loss function, such that

$$L_i(\theta_i) = E_{s,a\sim\rho(\cdot)}\left[(y_i - Q(s,a;\theta_i))^2\right] \tag{2.1}$$

in each iteration. Q-network estimated accurate action value within a finite state of action spaces, however, the learning dropped significantly with increasing action spaces. Deep Q-learning in a single-agent environment and finite action space such as Breakout and Enduro have achieved human-level performance [2]. However, more sophisticated algorithms are required for infinite or large action space environments for better learning of agents [2, 1]. Linear value iteration algorithms as an iterative update, where action value is estimated separately in each sequence, such that action value function $Q, a_t = max_a Q(\psi(s_t), a; \theta)$ was used with random weights of deep convolution network $Q$ was considered impractical.

Given the success of deep Q-Network as a function approximator [2] in finite action space, it was later studied and tested in continuous action spaces. The use of a deep neural network (DQN) as a function approximator in a continuous action space environment such as balancing cart-pole, pendulum swing, etc. [14] has proved to be more efficient than in a finite action space environment such as Atari games [2].

The development of reinforcement learning is continuously evolving with better and more efficient algorithms. DeepMind Alpha Go Zero [4] is a true pioneer in the field of reinforcement learning and artificial intelligence in general. This pioneering work demonstrated that pure reinforcement learning is feasible even in large action space environments to achieve superhuman-level performance [4]. Reinforcement learning combined with Monte Carlo Tree Search (MCTS) was implemented to achieve such performance that the Alpha Go Zero game defeated both Alpha Go and the human champion by a large margin.

## 2.2 Multi-Agent RL

Problem space that consists of multiple agents – multiple agents acting in the same environment, was later formulated as multi-agent deep reinforcement learning. The goal of MARL is to train agents so that each agent learns to make decisions independently or collaboratively. Training agents without being shared the knowledge of other agents were successfully implemented in path-finding problems and claimed to be useful for both cooperative and non-cooperative environments [20]. Unlike any classical single-agent reinforcement learning environment, formulating a multi-agent environment suffered from many challenges such as non-stationarity, sparse reward, communication, etc. Due to these challenges, non-cooperative learning was unable to converge as the size of complexity grows with each action step [1].

Following the success of Atari games [2], Egorov [8] utilized the same techniques to study mixed settings of competitive and cooperative environments by implementing pursuit-evasion games and demonstrated that generalization across environments

and varying numbers of agents in multi-agent systems as a remedy for challenges of classical value-based learning and has shown the possibility of using deep reinforcement learning [8]. Transfer learning was also introduced to speed up the learning and find local optimal.

In the most recent study of Deep Multi-Agent Reinforcement Learning for decentralized Continuous Cooperative Control, an extension of common discrete action multi-agent Q-learning (COMIX) and the factorization of joint Q-function (FacMADDPG) has been introduced in a decentralized and fully cooperative environment [19].

### 2.2.1 Collaborative MARL

Multi-Agent reinforcement has long and ongoing challenges when dealing with multiple agents. The problem of decision-making can be formulated in two ways. One in which each agent learns independently without any knowledge of other agents, also known as a competitive environment, to maximize their own reward. Whereas, in the other environment an agent shares the knowledge of the environment and learn to make decision collaboratively. In complex environment settings where the agents can not observe the full environment – problems we have defined where the agent (UAV) does not have the full knowledge of the environment, pose challenges such as partial observability, non-stationarity, and communications for collaborative decision-making. These problems have long been studied and have achieved astonishing milestones over the decades. In the early phase of MARL, it was deemed concluded that the agent performed well enough just by learning independently in both collaborative and non-collaborative environments [20]. However, these methods later proved inefficient as learning independently does not guarantee full convergence [1, 9].

Recently, an idea of communication among agents [9, 28] in order to maximize learning in the MARL environment has been proposed using reinforced and differentiable inter-agent learning. The study aimed to solve non-stationarity and partial observability problems has concluded communication among agents as the solution. In such problem space, centralized learning – having a common reward function; decentralized execution – independent Q-learning function have proved to be an efficient method [28]. However, in the fully observable environment, actor-critic-agent (A2C) based MARL outperforms the state-of-the-art MARL model increasing scalability and robustness [7].

### 2.2.2 Competitive MARL

Perhaps, the best-known example of a competitive multi-agent system is a multiplayer game in which the players compete against each other for example, Alpha Go, chess, etc. In this setting of problem space, the agent will learn to make decisions on its own without the knowledge of the complete environment and without the knowledge of other agents. The idea of competitive learning is to train agents to achieve some common goal before the other agents, such as winning the games by scoring higher points than other agents. The challenges of the MARL environ-

ment such as partial observability, scalability, etc. still exist in this type of problem domain. While some or all of these challenges were tackled by simply developing robust communication [9, 28] mechanism in cooperative MARL, this is not feasible in a competitive environment. A comprehensive framework for training competitive self-play agents known as TLeage [19] was developed in 2020 to overcome the non-trivial challenges of requiring huge datasets for training by implementing several mainstream competitive self-play (CSP) MARL algorithms.

### 2.2.3 Application of MARL

The horizon of the MARL application has grown significantly with improved methods. Even though the starting of RL was limited to relatively simpler problem space, this has been adapted in many industries. The biggest success of MARL and RL is in the gaming space with great achievements such as Alpha Go Zero, Atari, etc. The advancement of reinforcement learning not only exists in the gaming environment but has also extended its use case and development in many other industries. In recent years, autonomous and transport industries received much traction. The application of MARL in the autonomous industry includes self-driving cars, traffic light control, self-controlled robots, etc. While self-driving cars and autonomous robots are not new, RL has been successfully implemented in the traffic light control system [16] and cooperative lane-changing systems [29] of relatively complex mixed traffic environments. An A2C MARL [29] system has been developed for cooperative lane changing among the connected vehicles on the road.

Another application of MARL is within the problem space of resource allocation. The problem of dispatching jobs of n customers in k machines has been studied as a resource allocation environment using value-based function approximation [27], while a packet routing of wireless networks [3] formulated as MARL system uses Q-learning algorithms.

Interestingly, even with all the advancements in RL and MARL where many industries have received huge traction, the use of unmanned aerial vehicles in search and rescue has not been studied widely in the past. The study of reinforcement learning for coordination of UAVs using A2C [5] showed positive advancements in the field; however, the study conducted was broader in terms of problem areas ranging from disaster management, agriculture, defense, etc. Therefore, we see an opportunity to study this field further to verify previous findings and test its applicability in search and rescue tasks.

# Chapter 3

# Research Methodology

The study of MARL in a collaborative environment has been studied greatly and achieved phenomenal results in the past. This study aims to extend the power of MARL to search and rescue tasks using UAVs, where we formulate our research problems as complex multi-agent systems in a non-stationary, and partially observable environment. This section presents our research methodologies including philosophy, type, strategy, overall design, baselines, and evaluation metrics.

## 3.1 Research Philosophy

Our research goals and aims are greatly driven by the philosophical aspects of positivism within the scientific quantitative research paradigm. This is due to the nature of the study which aims to produce valid knowledge by experimenting with existing concepts or knowledge. Our research will extend the concept of multi-agent deep reinforcement learning that has been utilized in similar problem spaces and has achieved great results. The problem space of autonomous UAVs for search and rescue tasks has not been studied in the past - however, the experimental methods and knowledge of MARL exist. The hypothesis formulated in our study is that MARL can be applied in autonomous UAVs for search and rescue tasks while outperforming single-agent reinforcement learning. The MARL model will be trained and tested in the simulated environment and validate our hypothesis using multiple evaluation metrics.

## 3.2 Research Type

This quantitative study of MARL systems analysis begins with the knowledge and advancement of deep reinforcement learning to test and generalize the application of MARL in our research problem paradigm. The study begins with the theory of reinforcement learning and deep reinforcement learning that consist of multi-agents interacting in a complex environment. The knowledge and advancement in the field will be extended in our study to build a new MARL model and test our hypothesis using descriptive statistical evaluation metrics.

This study is of type deductive research due to the approach taken as top-down. The primary reason for deductive research is the complexity of the overall settings of research. Deductive research allows us to generalize our findings through statistical analysis and can be presented in graphs, plots, and charts [21]. Moreover, the data we are dealing with is mainly collected from the observation of training and testing evaluation metrics of our model and are in the form of numbers that can be easily compared with the baseline of our study and thereby allowing us to draw a conclusion.

## 3.3 Research Strategy

As outlined in section 3.1. the study naturally falls under the philosophy of positivism as the study aims to explore the problem based on the existing knowledge and development in reinforcement learning. The study also uses experiments as a research design where observation will be used as a data collection method. Therefore, the research will be conducted using a quantitative research strategy. The goal of the study - experimenting with the application of MARL systems design in search and rescue tasks using UAVs, aligns with the quantitative research strategy.

## 3.4 Research Design

This study of developing and testing cooperative MARL systems for their application in search and rescue tasks is designed in mixed of both experimental and descriptive statistical research design. The experimental part includes the development of a simulation environment where the MARL model has been trained and tested. The training and testing metrics are then collected using observation data collection methods for statistical analysis. The different components of our research design are detailed in the remaining section.

### 3.4.1 Environment Design

To best simulate the real-world scenario of search and rescue tasks using the multi-agent system, our environment setting must be complex enough. The simulated $10 \times 10$ matrix board contains different obstacles, that are positioned randomly. For an agent to search and rescue a human, the board also consists of a human at a random position as shown below in Fig 3.1. This information is modeled as the state or observation, which is fed into the neural network to predict the next best action to be taken by an agent. The shape of the state $S$ is of size (104, ), including the position of the agent, battery life, and a Boolean value of whether an agent is carrying any object or human such that,

$$S = [Board\ B_{10\times10},\ x,\ y,\ battery,\ object] \tag{3.1}$$

Where, the state $S$ is a flattened Numpy array of size (104, ), Board $B_{10\times10}$ represents each cell of the board which is initialized with the value of -2 for each empty

cell while random integer value for objects and humans. The value of the cell will be changed to -1 once the agent explores the cell, marking it as explored.

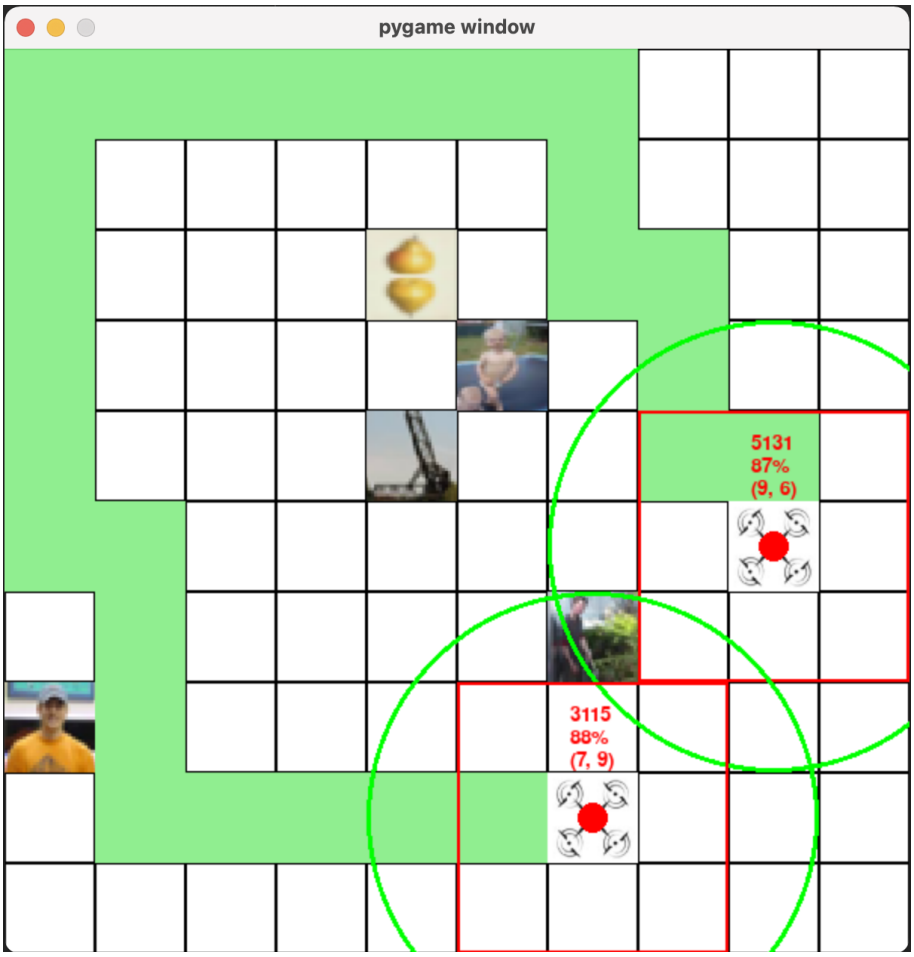


Figure 3.1: MARL search and rescue simulation environment. Obstacles and humans are placed at random positions. Two agents are displayed with battery info. The red squares are the observable environment by the agent, while the circles show the radius that the agent can communicate with other agents.

### The Agent

An agent (each UAV) has information such as agent ID, battery life, the radius at which it can share the information with another agent, and positions represented as coordinates x, y as shown in figure 3.1. The green circle (as shown in Figure 3.2) represents the proximity radius; if other agents appear within the radius the agents must exchange information with each other to increase the efficiency of search and rescue and avoid crashes. The red square represents the observation range for that particular agent, while green cells are the cells that are already explored by itself or other agents.

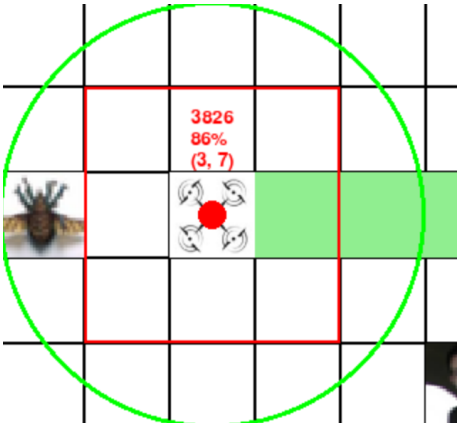


Figure 3.2: Single agent with associated information such as id (3826), battery life (86%), position on the board (3, 7).

### Action Space

In the Reinforcement Learning problem, defining actions that the agent can take is critical and it can be challenging based on the problem environment. For example, in the problem of the self-driving car, defining action is much more complex due to the large number of possible actions.

However, in our problem, we have defined a finite number of actions. The agent can take six possible action steps within the given environment. The agent can: move left, right, up, down, rescue, return to base, and do nothing. In reality, there may exist an infinite number of actions for the agents to perform better search and rescue, however, to simplify the training and testing we have defined discrete action space.

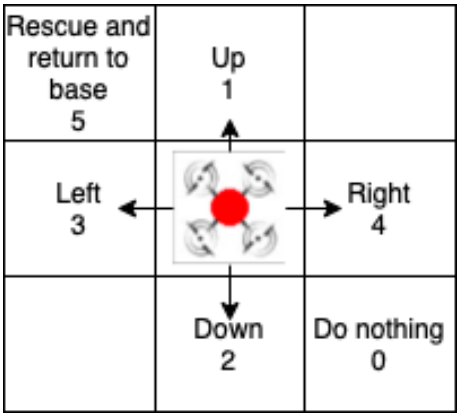


Figure 3.3: Possible actions that the agent can take in each time step $t$.

#### Communication

Knowledge sharing, also called communication, is the standard approach that stemmed from the recent advancements in MARL that centralized learning and decentralized execution to solve the existing challenges of non-stationarity and partial observability of multi-agent systems [9, 17, 28]. We have extended this idea to our experiments. The rule defined in our environment is simple, the agent must communicate with each other to share the full knowledge of the board they have gained when they are in a certain proximity range.

The communication range is defined based on the grid distance between each agent $A_1, A_2, A_3$. If the distance between two agents $A_1$ and $A_2$ noted as $d_a >= 5$, then both agent $A_1$ and $A_2$ are within the communication range of five grid distances and must exchange the map information with each other. The exchange of map information essentially exchanges the observation spaces including the known and unknown areas of each agent in the board. This ensures the coordination between the two agents to map the board effectively without having to explore the areas that have been explored by other agents.

## 3.5 Methods

As defined in section 1.4, our primary objective is to train multiple UAVs using reinforcement learning techniques to take decisions cooperatively to fully explore the unknown maps and rescue detected humans. We have formulated our multi-agent system as a cooperative decision-making problem for a team of UAVs deployed in the environment and trained to learn how to explore unknown areas of the map collaboratively using RL principle.

### 3.5.1 Reinforcement Learning

Reinforcement learning (RL) is a popular machine learning technique to train agents to perform a certain action in a way that will maximize the rewards [10]. This idea of maximizing rewards is then used for problem-solving. In our problem, the goal of all $N$ UAVs is to explore the entire map and rescue humans. The agents initially do not have access to any knowledge, they learn to maximize rewards that lead to efficiently exploring the entire search and rescue environment by exploration and exploitation strategy, known as the epsilon greedy strategy.

Each N agent takes action $a_t$ in the current state $s_t$, receives immediate reward $R_t$, and transitioned state $S\prime$ as shown in figure 3.4 while learning policy $Q(s, a)$ to predict the next action.

On a high level, the problem we are aiming to solve is based on the foundation of Markov's Decision Process (MDP), where the ultimate goal is to learn an optimal state value based on the action taken at a time step $a_t$ within the observed state $s_t$ and transitioned to a new state $s\prime$ by achieving some rewards $r$.

The goal of MDP is to find an optimal policy by taking any possible action in a given environment that guarantees maximum rewards. The optimal state value

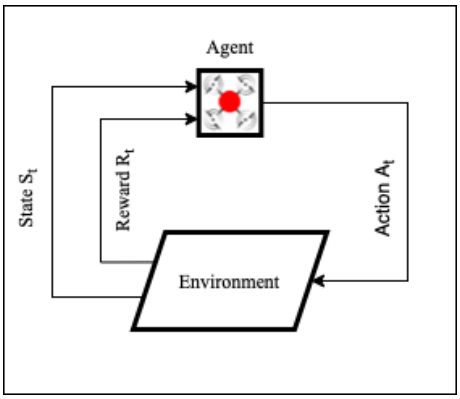


Figure 3.4: Reinforcement learning: Agent interacts with the Environment by taking an action $A_t$, receives immediate reward $R_t$ and transitioned into the state $S_t$

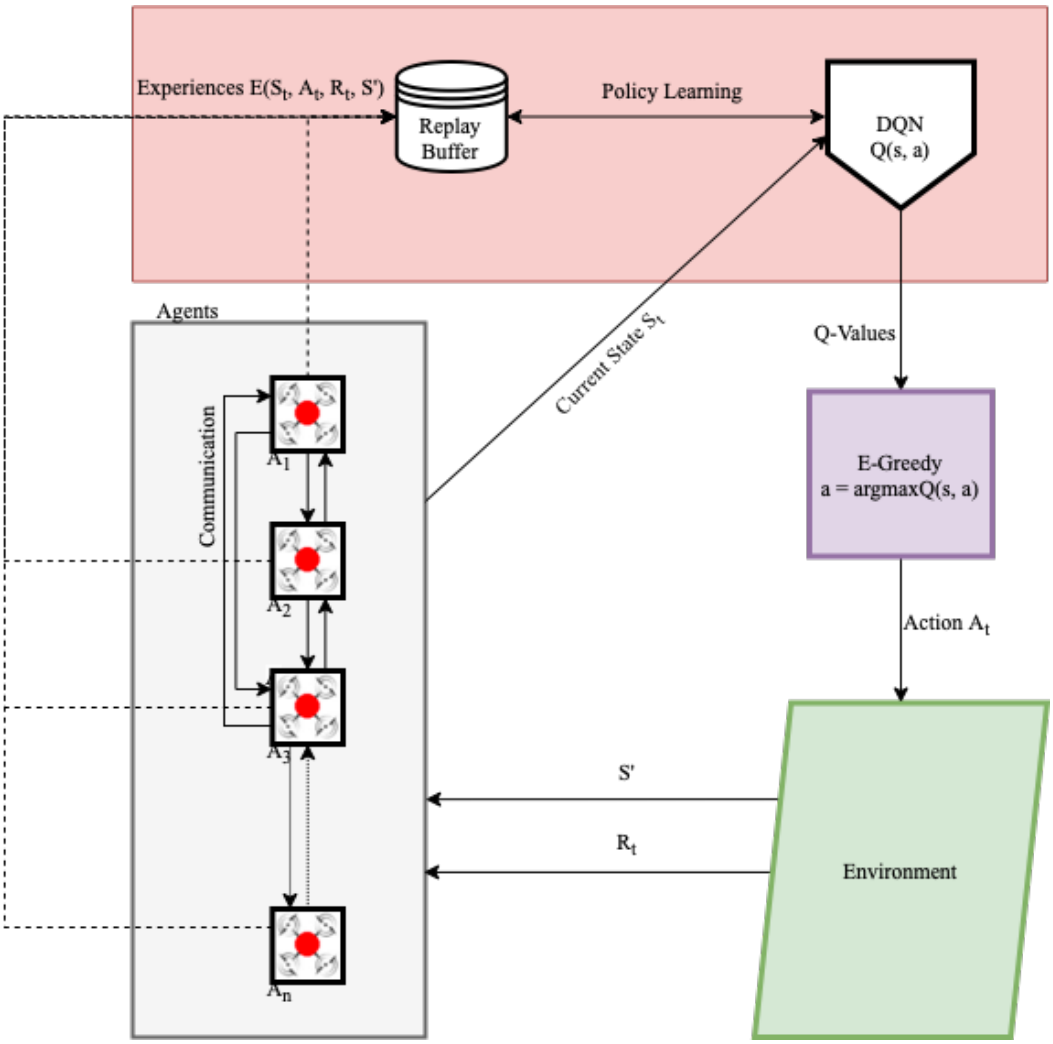


Figure 3.5: Multi-Agent RL architecture

is determined using Bellman Optimality Equation 3.2 [10] such that the optimal state value $V^*(S)$ is the sum of discounted future rewards expected from taking the action.

$$V^*(S) = max_a \sum_s T(s_t, a_t, s\prime)[R(s_t, a_t, s\prime) + \gamma.V^*(s\prime)] \tag{3.2}$$

Where, the optimal state value for all state S is $V^*(S)$ and is equal to the sum of cumulative reward $R(s_t, a_t, s\prime)$ given the transition probability $T(s_t, a_t, s\prime)$ for future state $V^*(s\prime)$.

In order for a policy $V$ to learn over time, the policy is updated iteratively in each time step using the value iteration algorithm defined in equation 3.3 [10].

$$V_{k+1}(S) \leftarrow max_a \sum_{s\prime} T(s_t, a_t, s\prime)[R(s_t, a_t, s\prime) + \gamma.V_k(s\prime)] \tag{3.3}$$

The estimated value of state s at the $k^th$ iteration is $V_k(s)$.

### Q-Learning

The value iteration algorithm defined above only learns the optimal state value, which does not provide great use cases other than evaluating policy. Our ultimate goal is to be able to train our agents so that they can make independent decisions; therefore, action values are critical aspects of our algorithm. The learning of policy to estimate optimal state-action values using a neural network as a function approximator is called Q-Learning [10], where the optimal values are known as Q-Values and are noted as $Q(s, a)$, and is a sum of all discounted future rewards expected by the agents after taking an action a, in a state s. The policy learning of optimal state-action values algorithm is an extension of value iteration algorithms with added state-action pair (s, a) and it is possible to approximate action-value using linear function approximator as shown in equation 3.4. However, it was more obvious in our experiment to choose a non-linear function approximator such as the neural network.

$$Q_{k+1}(s, a) \leftarrow \sum_{s\prime} T(s, a, s\prime)[R(s, a, s\prime) + \gamma.max_a Q_k(s\prime, a\prime)] \tag{3.4}$$

This algorithm gives us the Q-values for each action defined that can be taken in the given state. The choosing of the action by defining the policy $\pi^*(s)$, is done using the $\epsilon - Greedy\ Strategy$ [10],

$$\pi^*(s) = argmax_a Q * (s, a) \tag{3.5}$$

When the agent is in state s, it must choose the action with the highest Q-Values.

The policy network is then updated using the following equation,

$$Q(s, a) \leftarrow r + \gamma max_{a\prime} Q(s\prime, a\prime) \tag{3.6}$$

### Q-Network

Q-Network is a densely connected neural network, which takes observation at any given time step $O_t \equiv state\ s$ and outputs the Q-values corresponding to actions $a = [0, ..., 5]$. The neural network architecture generally consists of input, hidden and output layers. Our Q-Network consists of 128 input neurons as input layers, two hidden layers of shape 128 and output layers of shape 6.

Deep Q-Learning is a policy-based reinforcement learning technique, which directly searches over policy space using a deep neural network-based action value estimator [28]. Deep Q-learning uses the neural network as a function estimator and the rule is presented in equation 3.7, where weights $\theta$ of the Q-Network are updated using the loss function as shown in equation 3.8.

$$Q(s, a|\theta = Q(s, a|(\theta) + \alpha(r + \gamma maxQ(s\prime, a\prime|\theta) - Q(s, a|\theta)) \tag{3.7}$$

$$L(s,\ a|\theta_i)\ =\ (r\ +\ \gamma\ maxQ(s\prime,\ a|\theta_i)\ -\ Q(s,\ a|\theta_i))^2 \tag{3.8}$$

### Experience Replay

The learning of Q-Values is not as straightforward when the transition probabilities and rewards are unknown initially. In Reinforcement Learning, the agent learns by interacting with the environment directly and is called online policy learning, where, the policy $Q$ will be updated based on the agents' interaction and the return it received by taking an action in each time step. These methods often require many iterations before the agents efficiently learn to make decisions correctly and are much more time-consuming and memory inefficient.

To overcome these challenges experience replay [2] was introduced in 2013, where the policy $Q$ learns offline from its past experiences. In each iteration time step, a tuple of current state $s$, action $a$, reward $r$, and next state $s_t$ are stored in a centralized buffer of fixed length as a collection of experiences. A random sample of experiences is drawn from this buffer as a mini-batch and apply Deep Q-learning update as shown in equation 3.7 to update the policy $Q(s, a)$ at a specified interval.

In our multi-agent system, we have designed our system as centralized-learning-decentralize-execution, where there exists a single replay buffer of fixed size 50,000 to store the experiences $[s_t, a_t, r_t, s_{t+1}]$ of each agent $A_1, A_2, A_3$ and there exists a single policy $Q$ for learning. However, in each episodic time step, each agent makes independent decisions using the same policy $Q(s, a)$.

Table 3.1: Algorithm. Deep Q-Learning with experience replay buffer [13]

**Algorithm 1: Deep Q-Network with Experience Replay**

**Input:** Initialize replay memory $M$ with capacity $N$, exploration rate $\epsilon$, discount factor $\gamma$, and neural network with weights $\theta$

**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 $\epsilon$ 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**

Here, M is the replay buffer containing the experience of each agent in each time step $E = [s, a, r, s\prime]$, where $s$ is the current state, $a$ is action determined by the $\epsilon - Greedy$, $r$ is the reward gained and $s\prime$ is the state after taking action $a$. The exploration rate $\epsilon$ determines the probability of random action for exploration. The parameterized value of $\theta$ and $\theta_i$ is used for Q-value approximation of the target network and compute loss value of the target network respectively. The algorithm updates the target network based on the mini-bath of experiences sampled randomly from the memory M, in every time step $t = 5$ with online target network weights $\theta$.

### Reward Assignment

The reward $R$ in the Reinforcement Learning algorithm is the core foundation of the algorithm. As defined above, the reinforcement learning algorithms depend on how we define rewards in our environment - the performance of the algorithm can be significantly impacted by the reward assignment. Developing a sophisticated reward assignment is also very sparse depending on the tasks aimed to solve and the actions that the agent can take within the environment.

A positive or negative integer reward $R = -10\ to\ 100$, where -10 for invalid action, 10 for valid action, and a major reward of 100 for exploring the entire board and rescue, is given for each action $a = [0, 1, .., 5]$ taken in each time step $t = [1, ..., n]$

by each agent. Valid actions are defined as follows; if agent $A_1$ at position $(0,\ 1)$ determines an optimal action $a = 4 \equiv\rightarrow$ and moves one step right to the position $(1,\ 1)$, the action $a$ is considered valid if the cell at $(1,\ 1)$ was unknown - represented as $-2$ on the board and agent $A_1$ receives the positive reward of 10. However, if position $(1,\ 1)$ was already known - by $A_1$ or any other agents, or exists an object - represented by a random integer (97 *and* 47 as shown in figure 3.6), then the agent $A_1$ receives negative reward $-10$ and action $a$ is considered not optimal action.

Since the goal is to learn to make decisions collaboratively, each agent works towards the common goal, which is to maximize the global reward $R_g = \sum_1^n R_i$, where, $R_i$ is the episode reward of each agent and $n$ is the number of the iteration per episodes, and $i$ is the iteration.

### 3.5.2 Baseline and Evaluation Metrics

We have taken the performance of a single agent based on its learning metrics as the baseline for our multi-agent systems. The goal is to test the validity of our hypothesis that the multiple agents making coordinated decisions are likely to converge faster than the single agent acting in the same environment; therefore, deploying multiple agents in the field is more efficient than deploying a single agent.

While there are no single and better approaches to validate the metrics and conclude that multi-agent systems are performing better in a given scenario or the other way around; we have evaluated the rate of convergence based on the number of grid cells explored and the accumulated reward curve of both single and multi-agent systems. The key metrics evaluation questions are; how fast the convergence occurred? how smooth the accumulated reward and the accumulated number of cells curves were? how sparse was the reward accumulated over each episodic time step?

# Chapter 4

# Experiments and Results

In this chapter, we will present how we have conducted our experiments by outlining the experimental settings, and training parameters along with the key results and findings.

## 4.1 Experimental Setup

In the first phase of our experiments, we experimented our DQN algorithm for training a single agent, where one agent attempts to perform exploration and rescue tasks in the given environment. The training metrics - rewards $r$, number of cells explored $E$, and total steps taken $s$ were collected for each training episode $episode = 4,000$ of iteration $i = 1$ *to* $n$; $n$ is the last iteration when episode terminated, such that the episodic reward

$$r_n = r_{episode} = \sum_{1}^{n} r_i$$

and, number of cells explored per episode

$$E_n = E_{episode} = \sum_{1}^{n} E(i)$$

and total steps taken for the episode $s = n$. For better accuracy and reliability, we have calculated the mean value for every episodic epochs of $ep = 25$, for both accumulated reward and the number of cells explored for evaluation.

The collected metrics were then used as the baseline for our multi-agent system. The training parameters for all numbers of agents - up to 3 agents, were kept constant with that of single-agent; with added communication mechanism for collaboration, and the training metrics were collected using the same methods mentioned above for evaluation. The remaining sections describe our experimental setup for training agents.

### The Board

As discussed in section 3.4.1 and 3.5, the environment and DQN algorithm has been implemented using Python 3.8.5, Pygame, and TensorFlow. The simulation board environment, where the agents interact and learn, was implemented using a Pygame board of size $10 \times 10$ matrix. The board consists of three objects $O = 3$ and three humans $H = 3$ represented by a random positive integer, known and unknown areas of the board represented by -1 and -2 respectively. The board information is updated as the agents move from position $(x, y)$ to $(x\prime, y\prime)$. The position of agent $(x\prime, y\prime)$ is then marked as a known cell in the board.

### The Agent

The agent class is represented as the agent holding the information of a single agent. The agent was initialized at a random position $(x, y)$ in the board with a battery life of 100, agent name, observation range $O_r = 1$, and carry object to negative. While pre-trained object detection algorithms were used to detect objects and humans on the board, we have simplified agent detection by keeping track of the position of the other positions such that for agent $A_1$, a tuple of other agents position $[A(x, y), ..A_n(x_n, y_n)]$ is stored and updated in each action step. To best simulate the real environment, the battery life of the agent will decay by the factor of $decay = 0.5$ at each time step.

### The Network

The algorithm shown in Table 3.1, was implemented using the TensorFlow and Keras library. The DQN model is a neural network, consisting of a densely connected $128 \times 128$ input and hidden network with 6 output neurons, where input neurons correspond to the input - are flattened array of observed state and agent position, while output neurons correspond to the actions to be predicted. Two copies of the same neural network with weights $\theta = 0$ and $\theta_i = 0$ are initialized as training network and target network, where the training network was used for the prediction of Q-values, while, the target network was used as a function approximation for training network that updates the weight $\theta$ of training network with weight $\theta_i$ of target network at every time step $t_i = 5$, after learning from the random sample of experience using equation 3.7.

### The Replay Buffer

The experience replay memory $M$ was initialized with collection $deque(maxlen = 50000)$, which stores the last 50000 experience lists of tuples from all agents $A_n = N$, where experience $E = [(s_1, a_1, r_1, s\prime_1), ..., (s_n, a_n, r_n, s\prime_n)]$. The buffer is used for offline learning of the target network by sampling a random batch of experiences. Offline policy learning often requires extensive memory and is time-consuming. To overcome this challenge and avoid overfitting the network, we have used a batch size of 256.

**The State**

The state $s$, is the observed state of the board by the agent at each time step. The initial state $s$ observed by the agent $A$ can be represented as

$$s = [B_m,\ x,\ y,\ object,\ battery]$$

and are flattened array of shape $(104,)$, where, $B_m$ is the board map, and

$$x,\ y,\ object,\ battery$$

are the position of agent $A$, remaining battery life of agent $A$, and whether the agent $A$ is carrying any object. In each iteration $i$, the agent observes the state $s$, which is then fed to the DQN model as an input to predict the next optimal action value.

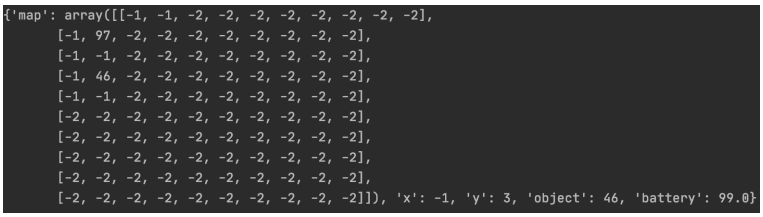


Figure 4.1: The state representation in the actual implementation

**Training Agents**

As described above, the agents have trained 4,000 episodes - one episode is considered complete after the termination of iteration; the termination condition can be one of the following 1) agent out of the board, 2) crashed with other agents, 3) out of battery, and 4) explored entire board (convergence). The complete training algorithm is shown in Table 3.1.

At the beginning of each episode, the agent first observes the state $s$ then $s$ is fed to the Q-Network to predict the action $a$ using the E-Greedy strategy shown in equation 3.5 and mapping the corresponding action with the highest Q-value. The agent will take predicted action and receives the next state $s\prime$, reward $r$, and determines if the state is in a terminal condition described above. The experience E will then be stored in buffer M. If M contains a number of samples equal to the mini-batch defined, the learning will begin by sampling random experience samples, and the Q-Network will be updated periodically using equations 3.7 and 3.8.

## 4.2 Experimental Results

We present our experimental results to show the capabilities of our MARL algorithm developed for the purpose of search and rescue tasks. Our results support our hypothesis and claims, which include 1) RL-based Multi-Agent Systems for search

Table 4.1: Hyperparameters used for the training of Agents

| **Hyperparameters** | **Value** | **Descriptions** |
|---|---|---|
| Replay buffer size | 50,000 | Memory size to store experiences |
| Number of agents | 1, 2, 3 | Number of agents experimented on separately |
| Learning rate ($\alpha$) | 0.9 | Learning rate for bellman equation |
| Discount factor ($\gamma$) | 0.8 | Discount factor used for Q-Learning update |
| Minibatch size | 256 | Experience sample for learning |
| Neural network hidden layers | 2 | Densely connected hidden layers |
| Hidden layer shape | 128 | Size of hidden network |
| Input layer shape | 104 | Size of input network |
| Output layer shape | 6 | Q-Values to predict (Actions to predict) |
| Mean reward epochs | 25 | Average reward epochs to smooth results |
| Number of objects | 3 | Objects placed on the board |
| Number of humans | 3 | Humans placed on the board |
| Optimization rate | 0.00025 | Optimization rate |
| Epsilon | 1 | Max epsilon for E-Greedy |
| Epsilon decay rate | 0.01 | Rate of epsilon decay |
| Training Episode | 4,000 | Agents were trained 4,000 episode |

and rescue task is possible, 2) Multi-agent learning outperforms single-agent learning, and 3) Collaborative decision-making is more effective than non-collaborative. To test our claims, we have trained our MARL systems for $n$ agents - both collaborative and non-collaborative. Collaborative learning is our core focus, where agents learn to make decisions cooperatively based on the rules specified in our communication mechanism. On the other hand, in non-collaborative learning, agents are unknown of the communication system; hence, they make decisions based on their own observations.

### 4.2.1 Single Vs Multi-Agents

The training parameters for the training of single and multi-agent are kept constant and the training metrics were collected. Both reward accumulation and board exploration rates of single agents - shown in Figure 4.2(a),(d), reached a peak by the learning episode of 1,500 and declined significantly until the learning episode of 2,700 before it starts to consolidate. Even though, the accumulated learning curve shows positive, the frequency of negative rewards and explorations are common in single-agent learning. Single-agent learning has achieved a maximum reward of 770 and a minimum reward of -560 while exploring a maximum of 79 and a minimum of 1 cell with a rate of exploration of 0.41960, where the rate of exploration is calculated as mean exploration divided by total grid size $b = 100$.

The reward accumulation during training multi-agent with constant training parameter as single-agent shows much stable reward accumulation as shown in Figure 4.2(b), (e), (c), and (f). The reward accumulation peaked at 500 episodes and maintained steady with minimal fluctuation. As shown in Figure 4.2(b) and (c), two-agent and three-agent respectively shows the mean reward accumulation fluctuated between 200 and 400. The actual minimum and reward received by two-agent and three-agent learning are -1150, 720, and -1350, 740. The exploration rate for both two-agent and three-agent are shown in Figure 4.2(e) and (f). Both multi-agent learning of two and three-agent successfully explored as high as 50 cells after learning episode of 1,000. The exploration rate for the two-agent slightly declined for the remaining episode of learning, however, the three-agent maintained a constant exploration rate with slight fluctuation. Our multi-agent system learned successfully to explore the environment with a rate of exploration of 0.40753 and 0.47859 with two and three-agent respectively.

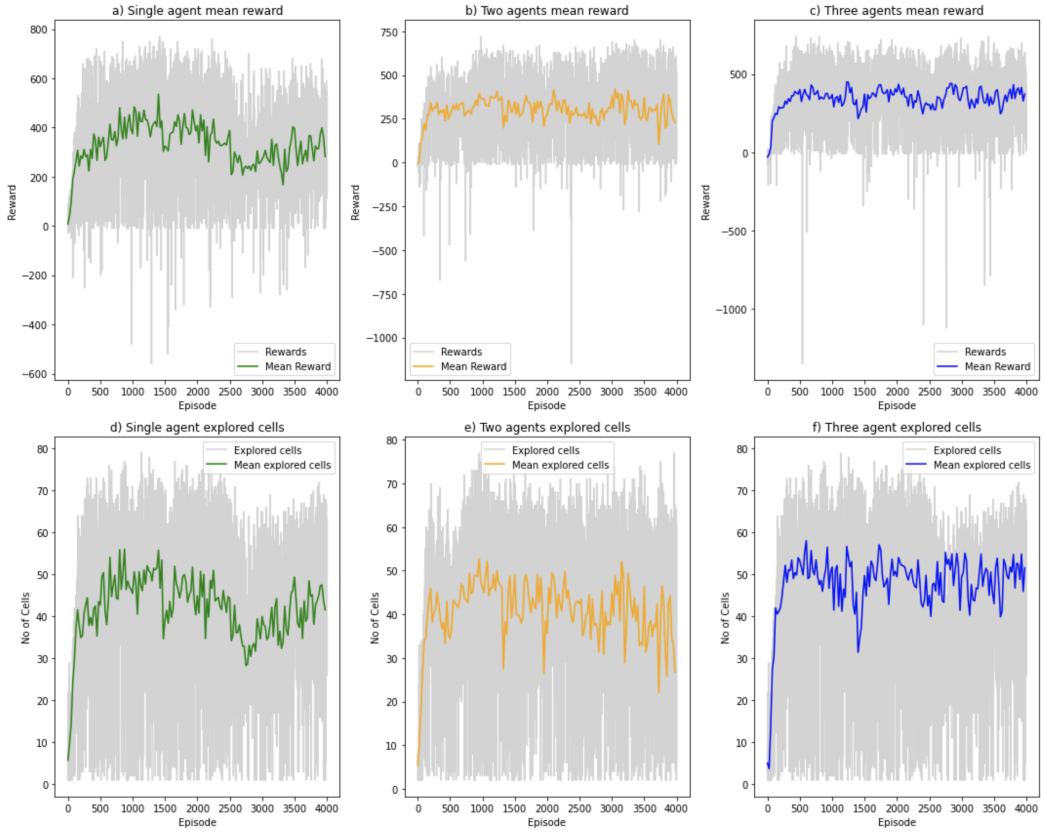


Figure 4.2: a) Single-agent rewards accumulation over 4,000 episodes, b) Two-agent rewards accumulation over 4,000 episodes, c) Three-agent rewards accumulation over 4,000 episodes, d) Single-agent board explorations over 4,000 episodes, e) Two-agent board explorations over 4,000 episodes, f) Three-agent board exploration over 4,000 episodes. Both reward accumulation and exploration are plotted on the y-axis against the training episodes on the x-axis.

The mean reward and exploration accumulated by our multi-agent training are

plotted against single-agent learning and are shown in Figure 4.3(a) and (b). Both reward and number of cells explored are the mean value of reward and cells explored at every 25 episodes. The mean reward accumulated by the single-agent exploring is significantly higher up to 1,500 episodes of training as compared to the multi-agent. However, this learning rate dropped by 70 percent 2,500 episodes onward. The multi-agent exploring environment accumulated much stable rewards throughout the training episodes. Training with two-agent performed poorly, while three-agent performed stable, and outperformed single-agent after the training episodes of 2,500.

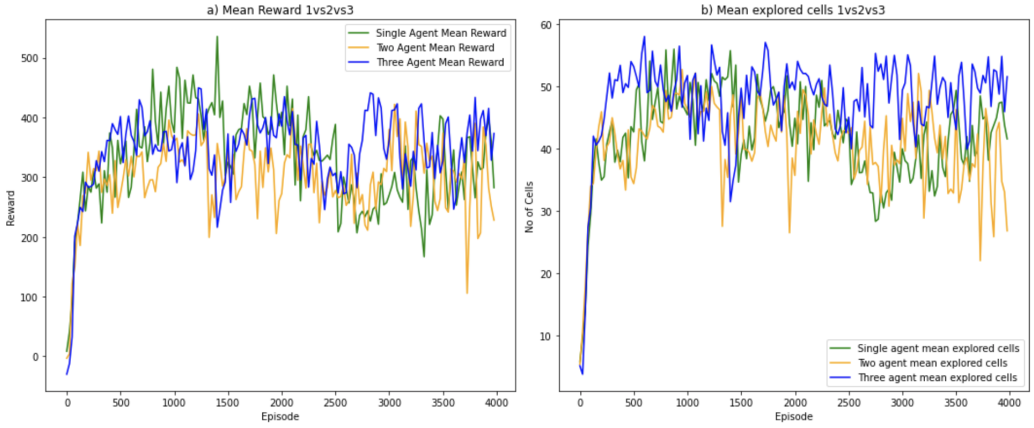


Figure 4.3: a) Accumulated reward - calculated mean reward for every 20 episodes for $n = 3$ agents, plotted on the y-axis, while episodes are plotted on the x-axis. b) Mean explored cells per 20 episodes of $n = 3$ agents learning are plotted on the y-axis, while episodes are on the x-axis.

In terms of exploring the board, our multi-agent has higher than the single agent. Training with two-agent has a similar exploration rate, while the performance dropped after the training episodes of 3,500 against the single agent. However, training with three-agent peaked at the training episodes of 500 exploring above 50 cells, and continued to explore for the rest of the training episodes.

The training with the single-agent fully converged - mapped the entire board, at the training episodes of 592 by taking 82 steps, while two and three agents exploring have converged at the training episodes of 278 and 282 taking 73 and 92 steps respectively and are shown in Table 4.2. The average reward received - for taking an action predicted by the $\epsilon - greedy$ strategy, per episode over the entire training steps, for one, two, and three agents is 330, 277, and 284 respectively.

### 4.2.2 Collaborative vs Non-Collaborative

Keeping the training parameters constant, we have trained two and three agents in our environment on two occasions - first with communication systems and without the communication systems in the second round. Collected training metrics - reward and cell exploration are plotted on the y-axis against the episodes on the x-axis, and are shown in Figure 4.4. As shown in Figure 4.4(a) and (b), the mean reward received

Table 4.2: Individual training metrics of agents 1, 2, and 3; where the training of agents 2, and 3 are collaborative

| **Agents** | **First convergence episode** | **Steps for convergence** | **Mean reward of learning** |
|---|---|---|---|
| Single Agent | 592 | 82 | 330 |
| Two Agent | 278 | 73 | 277 |
| Three Agent | 282 | 92 | 284 |

Table 4.3: Individual training metrics of agents 2 and 3; where the training of agents 2, and 3 are non-collaborative. NC refers to no convergence

| **Agents** | **First convergence episode** | **Steps for convergence** | **Mean reward of learning** |
|---|---|---|---|
| Two Agent | 607 | 174 | 293 |
| Three Agent | NC | NC | 307 |

while training two and three agents to explore the environment collaboratively is higher compared to the non-collaborative training. Training agents without the knowledge of other agents learned much slower than collaborative learning - the learning curve for non-collaborative is increasing over time, while the collaborative learning curve peaked and consolidates throughout the remaining training episodes. In order to reach the performance level of collaborative learning, non-collaborative learning required a significant amount of training.

On the flip side, the mean exploration of cells as shown in Figure 4.4(c) and (d), shows the opposite of reward accumulation. Here, non-collaborative learning has shown better exploration than collaborative learning. In both cases - two and three agents that are exploring the environment without communicating with other agents, the mean number of cells explored per episode peaked at 60 cells after the training of 500 episodes and has maintained this rate of exploration throughout the remaining training episodes. In comparison to collaborative learning, this is 16 percent higher at the same level of training. An additional 500 episodes of training were required for collaborative learning to match this performance. Although the learning trend for exploration is either increasing or fluctuating between 35 and 60, when two agents are exploring the environment collaboratively, the rate decreased after 3,500 episodes of training.

When two agents are exploring the environment non-collaboratively, it first converged after the training episode of 607 after taking 174 episodic steps as shown in Table 4.3. This number is significantly higher than agents exploring collaboratively, where it only took 73 episodic steps and converged at 278 episodes of training. In terms of reward accumulation, non-collaborative learning has a slightly higher accumulation - the mean reward of 293, while collaborative learning only accumulated a mean reward of 278 only. On the other hand, when three agents are exploring the same environment without communicating with other agents, it has never been able to map the entire board - no convergence at all; while the mean reward accumulated is still higher than collaborative learning. Since, each agent can only have their own observation - unable to see the full map - leading to the problem of par-

tial observability, non-collaborative learning performed poorly and never converged. When the communication system is on, the agents will share the map information including known and unknown areas of the board; therefore, each agent will have complete knowledge of the board. Our algorithm has performed well when the updated observation is fed, predicting the correct action values.

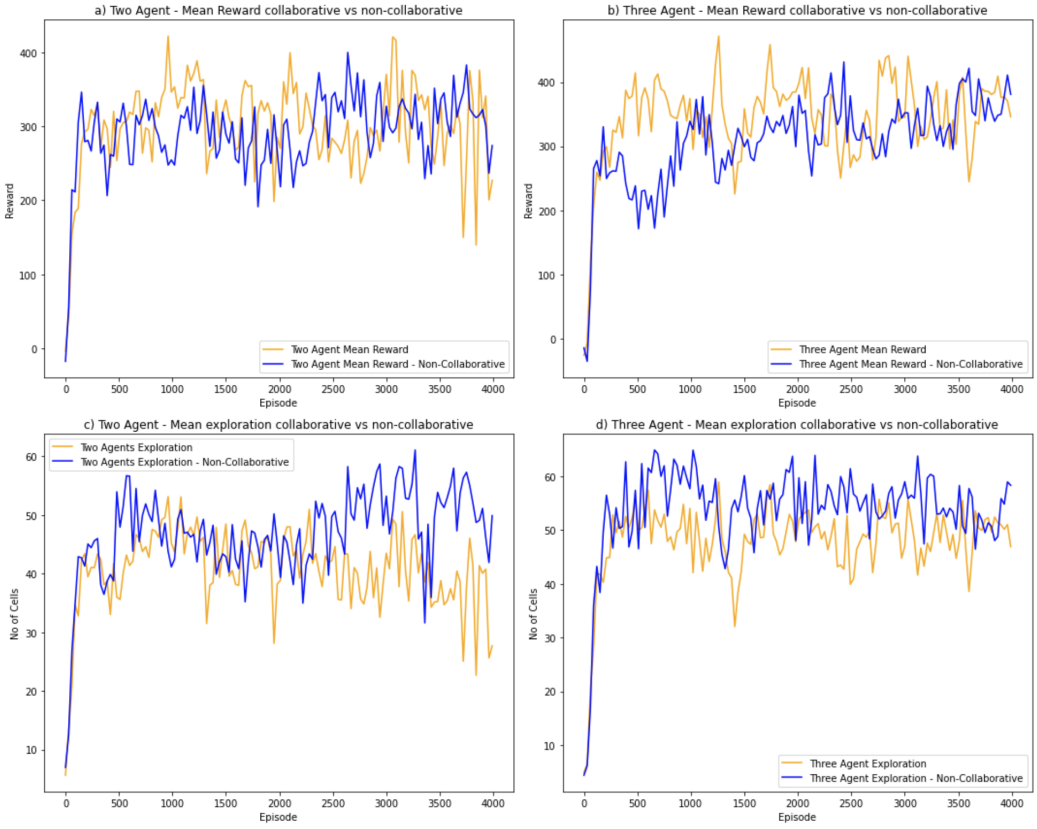


Figure 4.4: a) Accumulated Mean Reward - Two agents exploring the environment collaboratively and non-collaboratively are plotted on the y-axis, while episodes are plotted on the x-axis. b) Accumulated Mean Reward - Three agents exploring the environment collaboratively and non-collaboratively are plotted on the y-axis, while episodes are plotted on the x-axis.

# Chapter 5

# Discussion and Limitations

In this chapter, we discuss the findings and relevance of our training results outlined in Chapter 4 to our research aims and hypothesis. Furthermore, this chapter further explores the potential limitations of our research and expected future directions after this research to support our claims further and utilize our work.

## 5.1 Discussion

The primary objective of our research is to explore and experiment with the application of MARL in the search and rescue environment that utilizes UAVs to perform search and rescue tasks autonomously. In order to validate our research question - training UAVs using RL to make decisions cooperatively in a search and rescue environment, we have designed Deep Reinforcement Learning based algorithms and trained our agents within a simulation-based search and rescue environment. The training results of our algorithms are outlined in Chapter 4.

Developing an algorithm for autonomous UAVs to perform search and rescue is a novel idea in a disaster recovery problem paradigm. Our primary objective for the study in this problem space is to develop such an intelligent algorithm to train UAVs and we assume that when pre-trained UAVs are deployed in the search and rescue environment, they can perform search and rescue intended objects or humans. Based on the training results - in both cases single vs multi-agent (as outlined in section 4.2.1) and collaborative vs non-collaborative (as outlined in section 4.2.2), our deep reinforcement learning algorithm successfully converged; successfully searched the entire board and rescued detected human, after certain episodes of training within our simulation environment. The algorithm learned to explore the entire board in 592, 278, and 282 episodes of training when deployed one, two, and three agents respectively as shown in Table 4.2. These results indicate the possibility of its application in the search and rescue problem paradigm to use pre-trained UAVs - with correct hyperparameter tuning of the algorithm, to perform search and rescue tasks and it holds true for our hypothesis.

We have introduced an additional layer of abstraction known as collaborative search and rescue for better efficiency in completing tasks; therefore, we claim that multiple

agents performing search and rescue tasks collaboratively will have better learning performance. To test these claims, we have designed a simple communication mechanism that allows UAVs to share accumulated knowledge; the performance such as reward and cell exploration rate were collected for both single and multi-agent learning. Single-Agent learning fully converged in episode 592 of training for the first time (as shown in Table 4.2), while multi-agent learning converged in 278 episodes of learning. This indicates that the cooperative learning of multi-agent is likely to learn two times faster than single-agent learning. As discussed in the previous sections, when the communication mechanism is on, the agents share the knowledge that they have discovered with other agents that are within the radius specified. This means, our neural network will now have more information to predict accurate action values, hence, it learns faster and converges faster. Without the additional knowledge of the unknown area of the board, the algorithm predicts the action that may be valid based on its own knowledge, however, it may no longer hold valid as the other agent might have discovered that particular area in the board. Furthermore, the mean reward and exploration accumulation chart (as shown in Figure 4.3), shows a much more stable learning curve for multi-agent learning compared to single-agent.

## 5.2 Limitations

The study conducted to develop, test and validate a multi-agent reinforcement learning algorithm to train UAVs to perform search and rescue tasks is a complex task on its own and it required a high level of understanding and expertise in the problem paradigm. Our work may shed light on this problem space and encourage further study, however, it certainly has numerous limitations.

Due to time constraints, we simplified the problem paradigm and formulated a limited number of hypotheses to validate in order to complete the study on time. Our study tests the use case of MARL in the search and rescue space and validates our claims based on the training results obtained while learning agents. However, the actual performance of an algorithm has not been tested independently in a testing environment. While training results have validated our hypotheses, the testing metrics are critical to assess the actual performance of any machine learning algorithms. Another limitation of our algorithm can be proper hyperparameter tuning. We have only conducted a limited and manual hyperparameter tuning to obtain initial training metrics for our experiments. Proper hyperparameter tuning may increase the performance of the algorithm and have much better and more reliable training and testing results that increase the confidence of the algorithm.

The next important limitation of our study that is critical to be addressed for better performance of our algorithm is to design better reward assignments - reward assignment impacts greatly for any reinforcement learning algorithm and is the basic foundation of RL as the Q-Values are updated based on the reward received for any action. Our reward assignment only takes into consideration that the agent must always be within the $10x10$ board, must explore the entire board for a bigger reward, must visit the unknown board area only, and must have enough battery in order to receive the positive reward. The agent will receive a negative reward for every

other action taken that did not satisfy the conditions mentioned above. However, there are more cases when the agent neither receives positive nor negative rewards, for example, if agents find objects or humans to rescue; the agent must rescue them and return to the base; $x = 0, y = 0$ in the board position, and receive a greater reward for completing this task.

## 5.3 Future Directions

The search and rescue industry is a very sensitive and critical sector for any nation. Automating the search and rescue by leveraging advanced AI and Machine Learning techniques and tools can enhance productivity, reduce risk, and reduce the cost of operation. However, the implementation of automation at such a level requires a higher level of confidence in technology. Further study, testing, and validation are crucial to ensure the reliability and effectiveness of collaborative decision-making in UAVs for search and rescue tasks. Our study serves as a valuable baseline for exploring the capabilities of cooperative multi-agent systems. However, there are several limitations that need to be addressed to advance this field. Therefore, future research should focus on improving the experimental design, setting more refined goals and objectives, and developing better and more reliable Multi-Agent Reinforcement Learning (MARL) systems.

# Chapter 6

# Conclusion

In this study, our objective was to develop Multi-Agent Reinforcement Learning (MARL) systems and train Unmanned Aerial Vehicles (UAVs) to make collaborative decisions for search and rescue missions. We formulated our problem as the development of a Multi-Agent Deep Reinforcement Learning algorithm capable of training UAVs, and we successfully designed a multi-agent system for collaborative decision-making using a Deep RL-based algorithm, training the UAVs within a simulation environment.

Throughout our research, we collected and analyzed training metrics such as accumulated rewards, explorations, and convergence for each training episode. These results were crucial in verifying the validity of our hypothesis as presented in Chapters 4 and 5. Based on the results obtained, we can confidently conclude that the deep reinforcement learning algorithm can indeed be effectively utilized for the training of UAVs in the context of search and rescue missions. The experimental results align with the claims we put forth in the previous chapters. Specifically, our findings support the following assertions:

(a) The MARL system is highly applicable in search and rescue missions, demonstrating its potential for improving the efficiency and effectiveness of such operations.

(b) The implementation of a Multi-Agent system, as opposed to a single-agent system, leads to faster and more reliable decision-making processes. The collaborative nature of the MARL system harnesses the collective intelligence of the UAVs, enabling them to work together toward accomplishing the mission objectives.

(c) The integration of collaborative decision-making within the MARL system proves to be efficient and faster in comparison to non-collaborative decision-making. The collective intelligence and cooperation among the UAVs facilitate quicker and more informed decision-making processes, contributing to improved mission outcomes.

Our study has successfully addressed the research objectives by developing and training MARL systems for search and rescue missions. The empirical evidence obtained from our experiments supports the effectiveness of the deep reinforcement learning algorithm for training UAVs. Furthermore, our findings validate the applicability of the MARL system in search and rescue scenarios, highlight the advantages of a Multi-Agent approach, and emphasize the efficiency of collaborative

decision-making.

Moving forward, it is crucial to continue exploring and refining the MARL systems in order to address the limitations identified in this study. Further research and development efforts should focus on enhancing the scalability, adaptability, and real-world feasibility of the proposed MARL framework. Additionally, conducting field trials and collaborations with search and rescue organizations will provide valuable insights for the practical implementation and deployment of AI-powered systems in real-world scenarios.

By expanding upon our work and building upon the foundation established in this study, we can pave the way for the development of advanced and reliable AI-driven search and rescue systems that have the potential to save lives, reduce risks, and optimize resource allocation in critical situations.

# BIBLIOGRAPHY


[1] The dynamics of reinforcement learning in cooperative multiagent systems, 1998.

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

[3] A multi-agent framework for packet routing in wireless sensor networks. *Sensors (Switzerland)*, 15:10026–10047, 2015.

[4] Mastering the game of go without human knowledge. *Nature*, 550:354–359, 10 2017.

[5] Yoav Alon and Huiyu Zhou. Multi-agent reinforcement learning for unmanned aerial vehicle coordination by multi-critic policy gradient optimization. 12 2020.

[6] Lorenzo Canese, Gian Carlo Cardarilli, Luca Di Nunzio, Rocco Fazzolari, Daniele Giardino, Marco Re, and Sergio Spanò. Multi-agent reinforcement learning: A review of challenges and applications, 6 2021.

[7] Tianshu Chu, Jie Wang, Lara Codeca, and Zhaojian Li. Multi-agent deep reinforcement learning for large-scale traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, 21:1086–1095, 3 2020.

[8] Maxim Egorov. Multi-agent deep reinforcement learning, 2016.

[9] Jakob N Foerster, Yannis M Assael, Nando De Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning, 2016.

[10] By Aurelien Geron. *Hands-on Machine Learning with Scikit-learn, Keras, and TensorFlow*. O'Reilly, second edition, 2019.

[11] Sven Gronauer and Klaus Diepold. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, 55:895–943, 2 2022.

[12] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey.

[13] Yuxi Li. Deep reinforcement learning: An overview. 1 2017.

[14] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. 9 2015.

[15] Afshin OroojlooyJadid and Davood Hajinezhad. A review of cooperative multi-agent deep reinforcement learning. 8 2019.

[16] Peter Palos and Arpad Huszak. Relight-wctm: Multi-agent reinforcement learning approach for traffic light control within a realistic traffic simulation. pages 62–65. Institute of Electrical and Electronics Engineers Inc., 7 2021.

[17] Georgios Papoudakis, Filippos Christianos, Arrasy Rahman, and Stefano V. Albrecht. Dealing with non-stationarity in multi-agent deep reinforcement learning. 6 2019.

[18] Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V Albrecht. Comparative evaluation of cooperative multi-agent deep reinforcement learning algorithms, 2020.

[19] Bei Peng, Tabish Rashid, Christian A. Schroeder de Witt, Pierre-Alexandre Kamienny, Philip H. S. Torr, Wendelin Böhmer, and Shimon Whiteson. Facmac: Factored multi-agent centralised policy gradients. 3 2020.

[20] Sandip Sen, Mahendra Sekaran, and John Hale. Learning to coordinate without sharing information, 1994.

[21] L Karen Soiferman. Compare and contrast inductive and deductive research approaches, 2010.

[22] Peng Sun, Jiechao Xiong, Lei Han, Xinghai Sun, Shuxing Li, Jiawei Xu, Meng Fang, and Zhengyou Zhang. Tleague: A framework for competitive self-play based distributed multi-agent reinforcement learning, 2020.

[23] Richard S Sutton, David Mcallester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation, 1999.

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

[25] Jianhao Wang, Zhizhou Ren, Beining Han, Jianing Ye, and Chongjie Zhang. Towards understanding cooperative multi-agent q-learning with value factorization, 2021.

[26] Hua Wei, Guanjie Zheng, Vikash Gayah, and Zhenhui Li. Recent advances in reinforcement learning for traffic signal control: A survey of models and evaluation, 2021.

[27] Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. Building generalizable agents with a realistic and rich 3d environment. 1 2018.

[28] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. 11 2019.

[29] Wei Zhou, Dong Chen, Jun Yan, Zhaojian Li, Huilin Yin, and Wanchen Ge. Multi-agent reinforcement learning for cooperative lane changing of connected and autonomous vehicles in mixed traffic. *Autonomous Intelligent Systems*, 2, 12 2022.