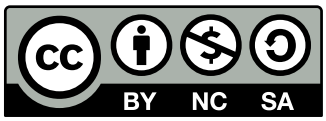
Aalto University
School of Electrical
Engineering

Master's programme in Machine Learning, Data Science and Artificial Intelligence

Multi-Agent Reinforcement Learning for Littoral Naval Warfare

Lauri Vasankari

Master's Thesis
2023



Aalto University
School of Electrical
Engineering

Author Lauri Vasankari

Title Multi-Agent Reinforcement Learning for Littoral Naval Warfare

Degree programme Computer, Communication and Information Sciences

Major Machine Learning, Data Science and Artificial Intelligence

Supervisor Prof. Joni Pajarinen

Advisor Prof. Jukka Heikkonen

Collaborative partner Department of Computer Science

Date 19.12.2023 **Number of pages** 92 **Language** English

Abstract

Littoral warfare refers to naval combat that occurs near coastlines or in seas with more constraints than the vast open oceans. Although the fundamental principles of naval warfare are consistent in littoral zones, this environment introduces unique challenges and demands specific tactics. Tactical doctrines must recognize the strengths and limitations of deploying forces in these areas, necessitating careful planning and decision-making. Decision-making in such scenarios is complex, requiring a balance between timeliness and thoroughness, often under conditions of limited resources and significant uncertainty. Machine learning has the potential to enhance decision-making in this context by predicting possible outcomes, detecting patterns, and determining optimal actions.

Naval conflicts have been infrequent throughout history, leading to a scarcity of data for applying data-intensive machine learning techniques. Naval forces consist of various units, and naval warfare is inherently competitive, involving two or more opposing sides. This environment is represented as a Partially Observable Stochastic Game (POSG), featuring two teams managed by decision-making agents. In POSG, the game is modeled through a framework of agents, states, actions, observations, and the probabilities of transitioning between states based on these actions. Multi-Agent Reinforcement Learning (MARL) algorithms are employed to devise policies that approximate solutions to POSG, aiming to address the challenges posed by this setting.

The study tests two MARL algorithms, Double Deep Q-Networks and Proximal Policy Optimization, within this multi-agent context. The findings suggest that despite complexities and the changing nature of the environment (non-stationarity), MARL approaches can generate and analyze different tactical options. This assists decision-makers by either reinforcing established tactical doctrines or offering new solutions, thereby enriching tactical planning and execution.

Keywords artificial intelligence, reinforcement learning, multi-agent, naval warfare, decision making

Aalto-yliopisto
Sähkötekniikan
korkeakoulu

Tekijä Lauri Vasankari

Työn nimi Multi-Agent Reinforcement Learning for Littoral Naval Warfare

Koulutusohjelma Machine Learning, Data Science and Artificial Intelligence

Pääaine Machine Learning, Data Science and Artificial Intelligence

Työn valvoja Prof. Joni Pajarinen

Työn ohjaaja Prof. Jukka Heikkonen

Yhteistyötaho Department of Computer Science

Päivämäärä 19.12.2023 **Sivumäärä** 92 **Kieli** englanti

Tiivistelmä

Rajoitetun merialueen sodankäynti viittaa merisodankäyntiin, joka tapahtuu rannikkoalueiden läheisyydessä tai valtameriä rajoitetummille merialueilla. Vaikka merisodankäynnin perusperiaatteet pysyvät samoina rajoitetuilla merialueilla, tämä ympäristö tuo mukanaan erityisiä haasteita ja vaatii erityisiä taktiikoita. Taktisten doktriinien täytyy tunnistaa toiminta-alueen vahvuudet ja rajoitukset joukkojen operoinnille, mikä edellyttää huolellista suunnittelua ja päätöksentekoa. Päätöksenteko on haastavaa, sillä se vaatii nopeuden ja perusteellisuuden tasapainottamista usein rajallisten resurssien ja merkittävien epävarmuuksien suhteen. Koneoppimisella on potentiaalia parantaa päätöksentekoa esimerkiksi ennakoimalla mahdollisia lopputuloksia, havaitsemalla kaavamaisuuksia ja määrittämällä optimaalisia toimintatapoja.

Merisodat ovat historiallisesti verrattain harvinaisia, mikä johtaa datan niukkuuteen dataintensiivisiä koneoppimismenetelmiä ajatellen. Laivastot koostuvat useista yksiköistä, ja sodankäynti on perimmiltään kilpailullinen, kahden tai useamman osapuolen välinen kamppailu. Tämä ympäristö voidaan nähdä osittain havaittavana stokastisena pelinä (POSG), jossa on kaksi joukkuetta, jotka ovat omien päätöksentekijöidensä hallinnassa. POSG:ssa peli mallinnetaan agenttien, tilojen, toimintojen, havaintojen ja tilanmuutosten todennäköisyyksien avulla. Moniagenttisen vahvistusoppimisen (MARL) algoritmit kykenevät oppimaan toimintamalleja, jotka approksimoivat POSG:n ratkaisuja. Näin ollen niitä käytetään tämän asetelman haasteiden ratkaisemiseen.

Tutkimuksessa testataan kahta MARL-algoritmia, Double Deep Q-Networksia ja Proximal Policy Optimizationia, moniagenttisessa ympäristössä. Tulokset viittaavat siihen, että huolimatta ympäristön monimutkaisuudesta ja muuttuvuudesta (epästationaarisuudesta), MARL-menetelmät voivat tuottaa ja analysoida erilaisia taktisia vaihtoehtoja. Tämä auttaa päätöksentekijöitä joko todentamaan vakiintuneita taktiikoita tai tarjoamaan uusia ratkaisuja, rikastaen näin taktista suunnittelua ja toteutusta.

Avainsanat tekoäly, vahvistusoppiminen, merisodankäynti, päätöksenteon tuki

Preface

I want to thank Professor Joni Pajarinen and my instructor, Professor Jukka Heikkonen, for their guidance and expertise, as well as Professor Kai Virtanen for his meticulous last-minute quality checks.

I also want to thank my offspring for their understanding, patience and relentless capacity to drag me from the abstract to the concrete.

Otaniemi, 16.12.2023

Lauri K. Vasankari

Contents

Symbols and abbreviations

Symbols

s, s'	state and succeeding state
S	set of all nonterminal states
S^+	set of all states including terminal states
$a \in A$	action in action space A
α	learning ratio
$r \in R$	reward in set of all possible rewards
$\hat{R}$	set of discounted rewards
t	discrete time step
L	loss value
π	policy
$\pi(s)$	action taken in s under deterministic policy
$\pi(a\|s)$	probability of taking action a under stochastic policy
$v(s)$	value of state s
V	array estimates of state-value function v
$q(s, a)$	value of taking action a in state s
Q	array estimates of action value function q
w	weight vector
$\mathbb{E}[.]$	expected value
λ	learning rate
σ	used to denote radar cross-section while σ^2 denotes standard deviation

Operators

$\frac{\mathrm{d}}{\mathrm{d}t}$	derivative with respect to variable t
$\frac{\partial}{\partial t}$	partial derivative with respect to variable t
$\sum_i$	sum over index i

Abbreviations

AI	artificial intelligence
ML	machine learning
MDP	Markov Decision Process
POMDP	Partially observable MDP
POSG	Partially observable stochastic game
RL	reinforcement learning
MARL	multi-agent reinforcement learning
DL	deep learning
DRL	deep reinforcement learning
CNN	convolutional neural network
RCS	radar cross-section
SSM	surface-to-surface missile
ASCM	anti-ship cruise missile, same as SSM but platform independent
PGG	fast attack craft, patrol gunship equipped with guided missiles
F2T2EA	find, fix, track, target, engage, assess
EMCON	emission control
DDQN	double deep Q-network
PPO	proximal policy approximation
MAPPO	multi-agent proximal policy approximation

1 Introduction

1.1 Background

Artificial intelligence (AI) is a cross-industrially trending technology that showcases performance and possibilities on many fronts, from civilian everyday life to advanced military technology. A book by Tangredi et al. [1] has examined the impact and applications of AI in the context of US Navy. An earlier thesis on the field of Military Sciences [2] has examined the possibilities and restrictions of applying AI in naval forces on a Finnish task group level, from improving sensor usage through fusion, amplification and automated object classification to supporting decision-making in a tactical framework. One of the key findings in both works was that some functions in naval forces produce vast amounts of data that merely need to be adequately harnessed to be leveraged. At the same time, many functionalities, such as tactical decision-making, do not possess vast amounts of data to exploit. Data on naval combat is mainly historical and anecdotal; the Falkland War [3] can be seen as the latest example of modern naval warfare before the current asymmetric naval warfare exhibited in Russia-Ukraine War [4]. Despite similarities and general principles that have been applied in naval combat for hundreds of years, it can be argued that the records do not possess data that could be used to apply AI to enhance modern operations in the littoral environment, which is the focus of this thesis.

Tangredi et al. [1] denote that decision-making support with AI consists mainly of enhancing the analysis that leads to decision-making, as AI solutions are, for example, able to process vast amounts of data. As researched in [2], supporting tactical decision-making with AI could happen either by optimizing one's own resources to a perceived, analyzed threat or by creating a simulator or a game that enables one to either test a plan of action against an estimated opponent's course of action or to solve a general optimization problem that could exhibit unforeseen tactical actions by either side. The optimal course of action could be utilized in planning one's own tactical manoeuvres.

A warfare game typically has more than one player or agent, usually divided into two or more opposing teams. Warfare includes considerable uncertainty from human factors to technical equipment and their reliability to the impact of weather and other natural phenomena, and additionally, the uncertainty of adversaries' actions and intent. These characteristics are coupled with partial observability: unlike a game of chess, the battleground or area of operations is seldom fully observable to the participants, at least continuously, and the capabilities and intent of the adversary may remain unobserved for the whole duration of the operation. Therefore, a suitable way to formulate a warfare scenario is a partially observable stochastic game (POSG) [5], a mathematical model extending Markov Decision Processes [6] to multiple agents and partial observability.

Reinforcement Learning (RL) is able to solve POSG [5] under certain conditions. Within the same game concept, RL has caused a stir in the AI community with the lead of DeepMind. The company created AlphaGo [7], an AI that was able to win the game of GO against world champion Lee Sedol. One particular move by AlphaGo

surprised Sedol [8] and seemed at first like a mistake, which can be seen as an example of capability not only to enhance previously adapted methods and procedures but to come up with profoundly new ways to solve problems, whether by accident or on purpose. Since AlphaGo, DeepMind has further expanded its scope and created AlphaStar [9], which has been able to rank among the top 0.2 % of human players in a real-time strategy game StarCraft II, using multi-agent reinforcement learning.

DeepMind has exhibited outstanding performance in applying machine learning and reinforcement learning to Go and StarCraft II environments, demonstrating AI ability in complex games, but modern militaries are lacking. This appears to be due to the lack of suitable environments, as the real world is highly complex, not least in the military setting. Regarding observability, Go is a game where both players have full situational awareness of their own and their opponent's situation, which is very different from a military conflict where each side prefers to conceal their actions or divert and distract the opponent. In this sense, the AlphaStar demonstrates a comparable performance, as StarCraft II requires the player to scout the opponent and make decisions based on partially observed data and uncertainties. There do exist environments that could be utilized in the military, such as *Command: Modern Operations*, which allows the simulation of "every military engagement from post World War II to the present day and beyond" [10]. The game has many units and features, but as such, requires a vast amount of work to be fitted to a particular scenario if used to support decision-making.

1.2 Earlier research

This thesis focuses on surface warfare in the littoral environment to demonstrate the use of a simplistic game environment and multi-agent reinforcement learning to solve a well-defined, enclosed warfare optimisation problem w.r.t. available resources. Previous research on the subject includes a salvo model by Wayne Hughes [11], in which the staying power of a surface action group is calculated in a deterministic manner as

$$\Delta A = \frac{\beta B - a_3 A}{a_1}, \Delta B = \frac{\alpha A - b_3 B}{b_1} \tag{1}$$

where A is the number of units in force A, B is the number of units in force B, β denotes the number of well-aimed missiles fired by each B unit and likewise α denotes the number of well-aimed missiles from each A unit. a_1 and b_1 denote the number of hits A and B units need to receive to be taken out of action, while a_3 and b_3 denote the number of missiles destroyed by A and B, respectively. The result gives the number of units taken out of action by each salvo. This model displays the principles of surface warfare, where the ability to cause damage is balanced by the ability to counter the engagement or by being able to sustain damage taken. It is evident that the reliability of the result depends on correct data or assumptions on the variables, which means that the one utilizing the formula would need specific information about the opponent's capabilities in countermeasures, damage control, and ship structure.

The salvo model has been researched and expanded on, e.g. with Michael Armstrong proposing a stochastic salvo model [12] to introduce uncertainties related to such models and examining effective salvo sizes [13].

As an example of quantified analysis on combat, Dupuy [14] created Quantified Judgement Analysis (QJM) model in 1974 to predict the outcomes of future battles based on analysis of historic battles, analyzing the parameters and factors that have influenced the results.

Reinforcement learning has been researched in the naval context by, e.g., Mooren [15] and Wang et al. [16], in more technically confined problems. Both cited works optimize certain warship functionality w.r.t. its goal or desired end state. Rempel et al. [17] have written a review of approximate dynamic programming applications within military operations research, which resembles reinforcement learning, viewing Markov decision processes in dynamic programming and approximate dynamic programming (ADP). The underlying logic is the same as in reinforcement learning, where the program aims to maximize the return reward through consequent actions. Rempel et al. have listed seventeen articles on ADP within military operations research from 1995 to 2021. The articles encompass application areas of investing and planning, force structure analysis, personnel sustainment, situational awareness, missile defence, air combat, battlefield strategy, airlift, weapon target assignment, combat medical evacuation, illegal fishing patrols, and inventory routing. Out of the reviewed articles, Sztykgold et al. [18] is closest to this thesis by subject, as they have examined assisting a military decision-maker in battlefield tactics by modelling a conflict as a game and using multi-valued graphs to find an optimal path which is most likely to result in a victory given the uncertainty related to enemy movements and actions. In brief, Sztykgold et al. have modelled the battlefield as nodes and vertices, nodes being locations. Armies can either move or fire, and the mission is represented by a location to be reached within a specific time while preserving some minimum strength ratio between friendly and opposing armies. Using an algorithm based on temporal difference learning, i.e. $TD(\lambda)$, the authors found experimentally that the algorithm returned an optimal control in more than 84 % of trials in a simple scenario of 16 vertices and a 1:2 strength ratio between enemies and allies.

Working on a similar topic, Kemple et al. [19] have experimented with littoral warfare simulation as a wargaming experiment. The purpose was to evaluate the performance of U.S. Navy Surface Action Groups (SAGs) against fast patrol boats (FPBs) in littoral waters. The experiment used human decision-makers and a *Battle Management Assessment System and Raid Originator Bogie Ingress* wargame, pitting a Blue Force against the Red Force consisting of FPBs. The difference in the approach in this thesis is the use of reinforcement learning to optimize the tactics between the opposing sides instead of human decision-makers.

1.3 Objective and purpose

As aforementioned, this thesis aims to demonstrate the use of a simple surface warfare environment and multi-agent reinforcement learning to solve a well-defined, enclosed surface warfare optimisation problem. The purpose is to test an easily deployable way

to utilize AI to support tactical decision-making in a computationally feasible manner that could be brought into a task group command level.

The objective is not to create an all-encompassing environment but rather a warfare branch-specific tool to compare and oppose existing tactical thinking in a specific task, in this case, surface warfare. As the environment simplifies the complex reality, taking into consideration only a portion of the affecting factors, it cannot be viewed as a decision-making tool in itself but rather a supporting model to test and compare plans of action.

As described in the previous section, earlier surface warfare models are crude and simplistic, such as the salvo model [11]. The idea behind the simplicity is to enable the use of such a model technically anywhere to estimate the outcome of a particular encounter. It is also of notice that the scope of the human deduction is limited and, in many cases, fixed to certain accepted principles that may further shrink creative thinking to an easily predictable, schematic process where things are done *as they always have been done*. With AI and RL, there is a possibility to challenge existing operational procedures to develop a novel idea to utilize one's resources optimally in a manner that is more difficult for the adversary to predict.

Another reason behind simple models is the acceptance that the real-world environment, especially in warfare, is so chaotic with an endless number of features and variables that formulating such an environment ideally is technically infeasible. It would make it computationally gruesome and most likely hinder the convergence rate of the solution so that it cannot be used in the planning schedule on the level intended, as task group staff usually plan their actions one to three days ahead. Hence, it is accepted that a crude model of the environment is used, selecting features and factors of most importance, aiming to stay true to Occam's razor principle that "plurality should not be posited without necessity" [20]. The feature selection is further elaborated in Section 4 on the environment.

In essence, the purpose is to display a way to formulate littoral surface warfare into a machine learning problem and to solve it, analyze the results and compare these to the existing prerequisites and principles regarding the subject. The results can either confirm the existing principles to work, propose novel solutions or prove unsuccessful in solving the environment, providing either insight into the current *modus operandi* or the need for further research on the subject.

1.4 Research question

The research question of the thesis is whether deep reinforcement learning can be utilized to support tactical decision-making in a littoral surface warfare setting at a naval task group level, using a simplified simulation of the actual operations theatre. The resulting subquestions are:

1. What are design choices for battle space representation that allow reinforcement learning w.r.t. existing principles of naval warfare?

2. Can discrete or continuous action space agents solve the chosen battle space representation?

3. If solvable, can the results enhance decision-making?

The research report is structured so that the second Section discusses AI and machine learning to accumulate basic knowledge of the field. Reinforcement learning is covered from Section 2.4 onwards, turning the scope from AI to RL and deep reinforcement learning.

After the essential theories and methodologies have been covered, the basic principles of naval warfare are explored to establish the essential mechanisms and variables that need to be considered to create an RL environment to test the hypothesis presented by the research question. Section 3 establishes the background for the first and second subquestions related to the battle space and identification of the most important aspects that need to be formulated into the environment.

Whilst the principles and boundaries of the environment have been determined, the environment is envisioned in Section 4. Two RL methodologies are used to test the hypothesis of using RL in said environment, using discrete and continuous action spaces to try different types of interaction mechanisms for different methods. Section 6 consists of the results of the tests and a discussion of the findings, answering the third subquestion and the main research question.

1.5 Research methodology

The research includes a literature survey into the theory of AI and reinforcement learning as well as naval warfare and tactics, to accumulate the needed theoretical basis for a functional simulation. Functional simulation is used to test the developed environment's logic and produce preliminary results of using RL to support naval commander decision-making in a computationally feasible, easily deployable manner.

As the topic resides in the military and the research problem requires formulation of the problem area and solving the problem, operations research (OR), also known as operational analysis, can be deemed a suitable framework for this thesis. Operations research has been defined as "A scientific method of providing executive departments with a quantitative basis for decisions regarding the operations under their control" [21]. With regard to the research problem, the quote seems fitting. Operations research in the military has its background in World War II when the idea of applying scientific methods to solve military operation problems was well established [22]. The fundamental question of a decision-maker that OR aims to answer is, "What course of action would be the best to achieve my objective?" This question requires an answer to several subquestions: What is the objective, what different courses of action are available, what factors (variables) are relevant for the course of action, and how is the effectiveness of a particular course of action measured.

The OR method consists [22] of the following steps:
Formulation

- Determination of the objectives of the operations involved
- Enumeration of all alternative courses of action

- Defining a measure of effectiveness by which to compare alternatives
- Determination and enumeration of the variables to be considered

Problem solution

- Theoretical analyses
- Statistical analyses of empirical data (data on operations already conducted)
- Controlled trials or exercises
- Simulation (experimental investigation using physical, analogue or digital models of the operational situation)

Communications of results

- Present results and conclusions for decision-making

In addition, mimicking cross-industry standard process for data mining (CRISP-DM) [23], when researching AI, the steps include *evaluation* of results before *deployment* for applications. Otherwise, CRISP-DM shares similarities with OR methods, as it includes deriving data understanding from business understanding, after which data preparation is executed to enable modelling before proceeding to evaluation and deployment. In OR, the business understanding is, in a way, included in the formulation. In contrast, the problem solution consists of the other steps before evaluation and deployment that are under communication of results.

This research will perform the formulation as described above but in problem solution steps focus on theoretical analyses, statistical analyses and primarily simulation with the developed game environment. No controlled trials or exercises are conducted.

Solving the problem in OR [22] generally consists of formulating or developing a mathematical model of a real-world scenario that represents or simulates the physical operation examined. Model building is a widely used scientific method outside OR. It is of notice that solving does not inherently mean that an optimal or any solution is found nor that a decision would automatically follow, but that the formulation can be solved to support the decision-maker by measuring the effectiveness of specific courses of action w.r.t. objective and determining their value under restrictions and limitations of the model. The input of OR is then valued against the subjective knowledge of the decision-maker so that the solution can enhance and supplement the final decision-making.

To follow the OR method steps, Sections 3 and 4 determine the objectives of the examined operation and identify and enumerate relevant variables to be considered in formulating the problem area. The measure of effectiveness is presented in Section 4 for the RL environment, in which the reward functions and winning criteria are determined. Demonstration of enumerating all alternative courses of action is done utilizing the multi-agent reinforcement learning framework to produce working policies, i.e. tactical approaches or courses of action.

For problem solution steps, theoretical analyses are done after receiving results from MARL solutions to assess the solutions' feasibility or reasons for not finding any. Before these, statistical studies of empirical data are used to establish the uncertainties related to the environment, i.e. to determine the probabilities for detection of targets and successful missile engagements. Despite being listed in the problem solution, these steps aid in formulating the problem into a mathematical model, which is then solved using MARL in the game environment.

For deep reinforcement learning, two methodologies are tested in the functional simulation, namely Double Deep Q Networks (DDQN) [24] and Multi-Agent Proximal Policy Optimization (MAPPO) [25].

1.6 Contribution

The main contribution of this thesis is to demonstrate the use of RL as a decision-making support tool for a tactical decision-maker. This is done by formulating the real-world problem into a digital environment and training intelligent agents to learn different policies.

The policies represent alternative courses of action that can be quantitatively evaluated based on the results. The idea is to enhance the planning process that leads to the decision-making by providing, with RL,

- Alternative courses of action
- Quantitative analysis of each alternative
- Opposition to planner and decision-maker bias
- Partial automation and speedup of a part of the planning process
- Possibly novel solutions to expand tactical thinking

A set of alternatives has to be drafted and analyzed to support the decision-maker. Hence, RL is used to learn policies that produce quantitative results w.r.t. the problem solution. These results can be used to analyze the alternatives and select the favoured course of action with a more robust idea of the likely outcome of that particular course of action.

The method is supposed to oppose planner and decision-maker bias by producing alternatives without the subjective, cultural mindset behind the analysis and favoured courses of action. As Colonel John Boyd has stated in his widely recognized OODA (Observation-Orientation-Decision-Action) [26] theory or model of a decision-making process that the *orientation* is "shaped by and shapes the interplay of genetic heritage, cultural tradition, previous experiences and unfolding circumstances". In other words, the decision-maker's personal traits play an undeniable role in the process. Therefore, an independent decision-making support system should be able to identify the non-favourable bias and likewise enforce objectively better solutions.

To the author's best knowledge, a similar study has not been conducted before in this context, and academically, this is the first demonstration of using RL in supporting littoral naval warfare decision-making.

2 Artificial Intelligence and Reinforcement Learning

2.1 Artificial Intelligence

There are many definitions for AI, but there is no global consensus on the meaning of the term. Russel et al. [27] have listed several definitions, of which the one by Bellman [28] is fitting for this thesis, as Bellman defines AI as "[The automation of] activities we associate with human thinking, such as decision-making, problem-solving, learning...". Elaine Rich [29] has defined AI as "the study of how to make computers do things at which, at the moment, people are better". These definitions highlight the goal of creating an artificial entity that can produce results similar to human thinking and acting. Russel et al. [27] provide a comprehensive and widely used source to further introduce oneself to the vast field of AI and its history, which is only briefly covered in this section. To remain explicit, in this thesis, the term AI is understood as Elaine Rich [29] had defined it and is described above.

According to Russel et al. [27], AI's philosophical foundations include using formal rules to draw valid conclusions, understanding the mind created by a physical brain, and exploring the origin of knowledge and its fruition to action. From a mathematical point of view, using formal, computable rules to draw conclusions and reasoning even under uncertainty stands out as most straightforward and parallel with formal science. It utilizes logic, computation and probability. Early formulations of logic are credited to George Boole in 1847, resulting in Boolean logic. Gottlob Frege extended the Boolean logic into first-order logic that remains used today. Kurt Goedel explored the limitations of first-order logic in his famous incompleteness theorem, which can be interpreted to show that some functions cannot be computed. Alan Turing created, along with the well-known Turing test that acts as a threshold of sorts between human and human-like intelligence, a thesis with Alonso Church which states that a Turing machine is capable of computing any computable function, also showing that there are functions no Turing machine can compute. A Turing machine [30] is a mathematical model of computation. The main takeaway regarding logic and computation in AI today is that the theoretical foundations have existed for an extended period. The research and advancement are incremental, with each breakthrough building on top of the foundations laid out earlier.

Like humans, an intelligent agent [27] has to handle uncertainty resulting from non-determinism, partial observability or a combination of these factors. The uncertainty in reasoning with logic and computation is formulated with the probability theory. The most notable rule is the Bayes' rule [27], invented by Thomas Bayes in the 18th century. The Bayes' rule can combine conditional evidence to scale up probabilistic systems with conditional independence, which allows the decomposition of large probabilistic domains into subsets, decreasing the growth of representation and computational complexity from exponential to linear.

The earliest work now recognized as AI was based on the physical brain and basic physiology of neurons, assuming that artificial neurons acted like switches, being either on or off, proposed by Warren McCulloch and Walter Pitts [27] in 1943. An activation would result due to the stimulation of a sufficient number of said neurons.

This structure could implement all the logical connectives of propositional logic with simple net structures. Marvin Minsky and Dean Edmonds built NARC, the first neural network computer, in 1950 using 3000 vacuum tubes to simulate a network of 40 neurons.

While the early onset focused on neural networks, the term AI was coined by John McCarthy [27] in 1956, and the general direction of the research was set by Allen Newell and Herbert Simon, who had created the "Logic Theorist", a reasoning program that was claimed to think non-numerically. Despite sharing similar goals, AI was also established as an independent field, separate from decision theory, operations research and control theory. It was also separated from mathematics, as AI is a branch of computer science and deals with human faculties in an interdisciplinary manner.

The research was focused on the logicist approach of McCarthy and symbolic AI [27] with Newell and Simon formulating the physical symbol system hypothesis, which states that "a physical symbol system has the necessary and sufficient means for general intelligent actions". This meant that any intelligent system must operate based on a system composed of symbols. Knowledge-based systems, where the knowledge base of an expert would be transformed into a program logic as an expert system to exhibit expert reasoning in a specific domain, were the first AI solutions to be industrialized.

The neural networks made a comeback to the forefront in the 1980s after several groups reinvented the back-propagation algorithm originally invented in 1969 by Bryson and Ho [27]. AI discipline had also isolated itself from existing fields between the 1950s and 1980s, but this approach was now abandoned to adopt scientific methods to advance the development further. Probability and decision theory were adapted to induce the above-mentioned ability to reason under uncertainty, and the Bayesian network was invented to enable efficient representation and reasoning with uncertain knowledge. In the era of the Internet and big data, subfields of AI, such as machine learning and data mining, have emerged. The topic of machine learning is disclosed in the next section.

2.2 Machine Learning

As opposed to logicist and symbolic approaches to AI [27], machine learning (ML) can be regarded as a numeric or connectionistic alternative. It should be noted that these schools of thought do not need to be regarded as competitors, but the main ideological difference between the early ideas of AI development and ML is that the end goal of developing knowledge bases was to create a general thinking machine, while ML aims to solve a particular problem by finding a working solution from available hypothesis space. It can be stated that ML is predictive, whereas statistical analysis is descriptive, although the definitions overlap at times: supervised methods can be seen as predictive, while many unsupervised methods provide insight into the data and can be regarded as descriptive. These terms are elaborated on below. In essence, ML [27] denotes an artificial agent's ability to learn and recognize patterns in data as well as to adapt to new circumstances. Goodfellow et al. [31] state that a "machine learning algorithm is an algorithm that is able to learn from data". The learning performed by the algorithm

in a program comprises data, a model that produces the performance by mapping the data into an inference, e.g., classifying a data sample to a certain class, and a function that evaluates this performance. The output of the evaluation function, which is the difference between the desired, actual output and the predicted output, is regarded as loss. Hence, a machine learning problem has three main components [32]: data, a hypothesis space of applicable models and a loss function. This can be formulated [32, 27] as

$$y \approx h(x),$$

where $y \in \mathcal{Y}$ is the label of the data point and $x \in \mathcal{X}$ are the features of the data point, with features $x = [x_1, x_2, ..., x_n]$. Hypothesis $h \in \mathcal{H}$ is one model or map in the hypothesis space $\mathcal{H}$ that maps the data point as $h(x) = \hat{y} \approx y$. The loss function is then used to calculate the error between $\hat{y}$ and y using, for example, mean squared error (MSE)

$$L = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2.$$

Based on the error, the hypothesis h is updated to decrease the error of the inference by tuning the parameters of the model or map, as usually $h = [w_1, w_2, ..., w_n]$, having a weight for each feature in x.

Depending on the available data and desired result, machine learning can be categorized into supervised, unsupervised and semi-supervised learning as well as reinforcement learning [27, 31]: if the data includes labels for the data points and these are utilized to fit the hypothesis to map the data to the labels, the learning task can be considered supervised as the evaluation is based solely on the difference between the predicted output and actual label of the data.

In unsupervised learning, there are no labels to be used for evaluation, and other metrics are used to execute inference on the data, extract information and discover hidden patterns. These are usually data compression or clustering methods that aim to group data points with some similarity metric, and the evaluation is done based on, e.g. Euclidean distance between features of data points. Common unsupervised algorithms [31] include k-means and Principal Component Analysis (PCA). PCA compresses data to a specified number of components by learning an orthogonal linear transformation of the data that projects the input x into a representation z by finding a linear transformation $z = W^\top x$ where the variance of x, denoted as $Var[x] = \frac{1}{m-1}X^\top X$ is diagonal. The problem can be solved using eigendecomposition, as the optimal $W_n = [W_1, W_2, ..., W_n]$ are the n eigenvectors of $X^\top X$ that correspond to the n largest eigenvalues. In the k-means algorithm [33] the sum of squares of Euclidean distances of data points, $Dist(x_i, y_j) = \|x_i - y_i\|_2^2$, where $\|\cdot\|_p$ is the L_p norm and $Dist(x, y)$ is the squared error of approximating a data point with its closest representative, is used to assign cluster labels for data points, thus providing insight to the data and existing relations of data points within.

In short, unsupervised learning can be regarded as a tool of data mining [33], which is the study of collecting, processing, and analyzing data and gaining useful

insights into it.

In semi-supervised learning, some data points are labelled, which can be used to enhance the inference from the unlabelled data points. Reinforcement learning means that the agent learns from a reward or penalty it receives from the actions it infers. The categorical distinctions are not as definite in practice as semi-supervised learning shows.

In addition to evaluating the output (criterion), the learning in each category requires a mechanism to update the parameters to decrease the loss or increase the reward. This is regarded as optimization [31]. If we have a function $y = f(x)$ where both $y, x \in \mathbb{R}$, the derivative of the function $f'(x) = \frac{dy}{dx}$ gives the slope of $f(x)$ in point x, which is used in said optimization to make a change in x for a small improvement in y by taking a small step into the opposite direction of the derivative. Functions with multiple inputs require partial derivatives, which give the impact of change for each weight parameter on the outcome and thus allow updating each weight parameter according to its impact on the criterion. This optimization is known as *gradient descent*, as the gradients are used to update the weight parameters downslope towards a smaller loss. The update rule can be formulated as

$$w' = w - \lambda \nabla_w f(w),$$

where w' is the updated weight vector, w the current weight vector and $\nabla f(w)$ is the vector of partial derivates. The learning rate, denoted λ, specifies the update (gradient) step size to avoid too large or minor changes in the weight parameters and thus allow the algorithm to find the minimum loss. The version known as *stochastic gradient descent* utilizes a random sample batch of the data. It updates the parameter weights according to the gradients in the said batch, which is assumed to represent the distribution of the whole data set. The stochastic version allows for more rapid updates as it bypasses the need to evaluate the gradients on the whole data set on all iterations as the updates are done in mini-batches throughout the *epoch*, which consists of the whole data divided into the mini-batches.

2.3 Deep learning

Deep learning is an increasingly important and one of the most successful subfields of machine learning, which utilizes artificial neural networks as the hypothesis space. The shortcomings of previously described, i.e. *conventional* or *traditional* machine learning algorithms in problems such as object and speech recognition have motivated the research and deployment of deep learning algorithms. The issue of conventional algorithms lies in the dimensional complexity of such problems, in which conventional machine learning fails to scale and generalize well. The networks utilized in deep learning increase the hypothesis and parameter space to fit and generalize well into such tasks as computer vision and natural language recognition. As such, a neural network can be regarded as an *universal approximator* [34], as a neural network are able to approximate any measurable function. Below, the key concepts of deep learning relevant to this thesis are briefly introduced.

Deep feedforward networks, also known as multi-layer perceptrons (MLPs) [31] are the common deep learning models with the goal of approximating some function f so that $y = f(x; \Theta)$ in a similar manner as supervised machine learning, only utilizing a multi-layer perceptron structure to learn the value of the parameters Θ that result in the best function approximation. A single perceptron [35] is visualized in Figure 1.

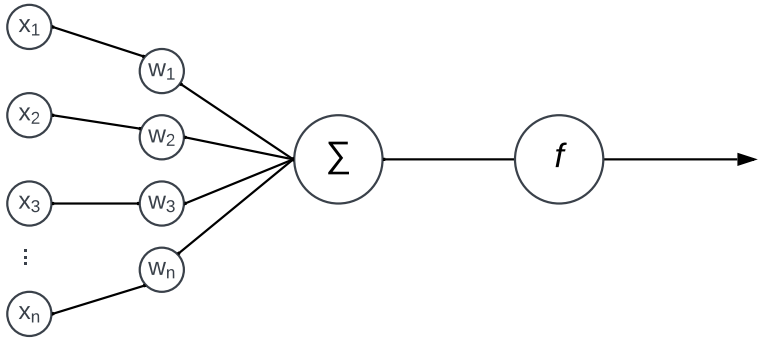


Figure 1: Perceptron [35]

In essence, the perceptron is a function that maps a real-valued vector to a binary output value and can be formulated as

$$f(x) = \left\{ \begin{array}{ll} 1 & \text{if } w \cdot x + b > 0, \\ 0 & \text{otherwise} \end{array} \right\},$$

where $w = [w_1, w_2, ..., w_n] \in \mathbb{R}^n$ and $w \cdot x$ is the dot product of the weight and feature vectors. Multilayer perceptron refers to composing a network of functions similar to the perceptron but in greater quantities, usually divided into subsequent layers. Given three functions $f^{(1)}$, $f^{(2)}$ and $f^{(3)}$ the network can be denoted as

$$f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x))),$$

where $f^{(1)}$ is the first layer of the network and so on. The depth of the network is the length of the chained functions, hence giving the name deep learning.

Due to the complexity of neural networks, the back-propagation algorithm [36] is used to calculate the gradient to use for model parameter updates. Back-propagation requires a computational graph [31] where each node indicates a variable for which the chain rule of calculus is computed in an efficient order. In vector notation, the generalized chain rule can be written as

$$\nabla_x z = \left(\frac{\partial y}{\partial x}\right)^{\mathsf{T}} \nabla_y z,$$

if $y = g(x)$ and $z = f(y)$, $x \in \mathbb{R}^m$ and $y \in \mathbb{R}^n$. The $\frac{\partial y}{\partial x}$ is the $n \times m$ Jacobian matrix of g, and the gradient of variable x is the multiple of the Jacobian matrix and the gradient $\nabla_y z$. Back-propagation performs this multiplication for each operation in the computational graph. Given four nodes $\{w, x, y, z\}$ in a graph, where $f : \mathbb{R} \rightarrow \mathbb{R}$ and the computational chain is $x = f(w)$, $y = f(x)$, $z = f(y)$, the partial derivatives $\frac{\partial z}{\partial w}$ can be obtained as $f'(y)f'(x)f'(w)$. A dot product between the gradient already

computed to child nodes of the current node and the partial derivatives of the same child nodes is also performed and added to the value. This way, back-propagation computation proceeds through the computational graph in reverse order to the original network graph. The updates on parameter values are done according to the gradient descent.

A conventional feed-forward neural network or MLP consists of only linear layers. The input is processed in a linear manner (forward propagation) through the network to produce the output. Hence, the input has to be of the size of the input layer, i.e. a $v^d, d = [2, n]$ tensor of inputs has to be flattened to have only the batch size and the 1D feature input. The MLP is not good at extracting meaningful patterns when processing grid data such as images. Pattern and feature recognition requires processing the image data to a more dense and feature-rich form. Therefore, image processing, among other higher dimensional grid data, is usually processed with convolutional neural networks that utilize the convolution operation

$$s = (x * w),$$

where x is the input and w is the kernel. If the input is two dimensional $x \in \mathbb{R}^{i \times j}$ the kernel $w \in \mathbb{R}^{2 \times 2}$ would result in a convolution operation as $x_{[0,0]}w_{[0,0]} + x_{[0,1]}w_{[0,1]} + x_{[1,0]}w_{[1,0]} + x_{[1,1]}w_{[1,1]}$. The output is the first pixel or point in the resulting mapping. The same computation is performed for the rest of the input in accordance with a predetermined step size. 1-D convolution can also be performed on one-dimensional data, such as sequential signals, if seen fit.

A common structure [31] of convolutional neural networks (CNN) includes layers that consist of the convolution operation or several consequent convolution operations, a nonlinear activation function such as ReLU (rectified linear unit) and a pooling layer that processes the output further, usually highlighting most prominent features. In the case of max pooling, the maximum value within the pooling kernel is selected for the output. The pooling layer makes the feature representation invariant to small deviations in the input and decreases the size of the output in relation to the size of the pooling kernel. After these stages, the operations may be repeated to process the features further. Finally, the CNN structure usually has a linear layer or layers that produce the outputs, e.g., to decide the classification of the processed image.

The third widely used family of neural networks are recurrent neural networks (RNN) [31], which are used to process sequential data such as natural language or signals through recurrence. The recurrence is implemented using a hidden state $h^{(t)}$ which takes into account the t previous time steps in the input to produce the output, i.e. $h^{(t)} = f(h^{(t-1)}, x^{(t)}; \phi)$ where ϕ are the parameters of the model and x is the input of some length. The RNN can be used for inputs of varying lengths, whereas feed-forward networks and CNN require fixed sizes of inputs.

2.4 Reinforcement Learning

In essence, while previously covered supervised and unsupervised learning aim, as a simplification, to either map data to a label or to extract information from the data, reinforcement learning [6] maps situations to actions in order to maximize a separate, numerical reward signal. The term RL denotes the field of machine learning, the problem, and the class of solution methods. The formulation requires an agent that interacts with an environment. The environment comprises everything outside the agent, and the interaction is continuous so that the agent's actions receive a response from the environment.

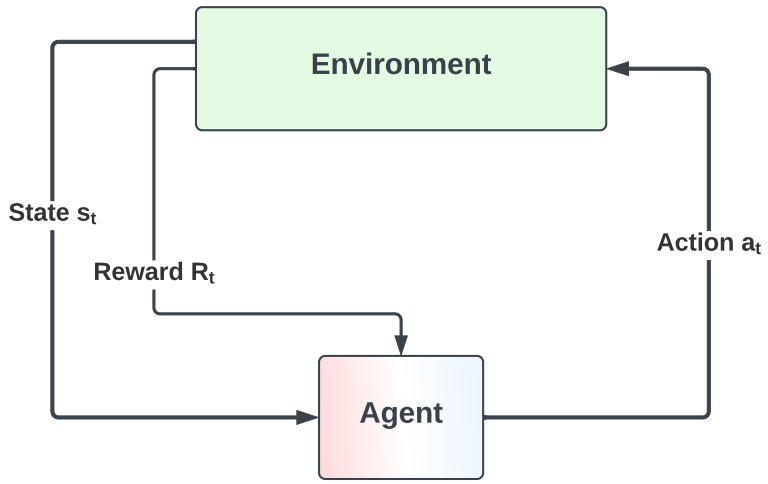


Figure 2: Schematic view of a reinforcement learning model w.r.t. Sutton et al. [37]

In RL, the agent learns [6] by accumulating experience in the particular environment by exploring different situation-action pairs w.r.t. respective reward. The environment can be regarded as static or dynamic based on whether it changes over time. A dynamic environment requires the agent to adapt to the changes accordingly. After sufficient exploration has been conducted, the agent can exploit the experience gained through exploration. The trade-off between exploration and exploitation is a problem particular to reinforcement learning, as the agent aims to learn a policy that optimizes the agent's behaviour in the environment by favouring tried-and-true actions for certain situations that maximize its reward. To find these actions, it has to try out actions it has not tried before. Sutton and Barton [6] also highlight that RL differs from other machine learning in considering the whole problem with a clear goal of performing well in that particular environment, whereas other machine learning subfields may produce results without knowing the end goal, i.e. supervised method can learn to classify some data to n categories, and this performance can serve in some other problem as a sub-solution. However, in RL, the solution is the solution for the whole problem without a division into sub-solutions.

Whereas machine learning can be considered to comprise three components (data, hypothesis space and loss function), RL can be seen to encompass four main elements [6]: a policy, a reward signal, a value function and (optionally) a model of

the environment. The policy denotes the agent's ability to map the perceived situation or environment state to the following action. The actions taken in the environment result in the reward signal, which acts as a way to measure the immediate result of the action taken w.r.t. the state perceived. The value function determines the goodness of the action in the long run, accumulating and estimating the following rewards from the current state onwards. Values are secondary to rewards, as the values represent cumulative rewards on a certain trajectory and thus do not exist without the rewards themselves. Despite being secondary to rewards, the values allow the agent to learn the optimal policy as the higher values represent the higher reward in the long run. The fourth element, the model of the environment, is essential for model-based methods where the models are used to plan future actions in the environment the models mimic. Consequently, model-free environments are simpler, and the learning is based solely on trial and error without planning.

2.5 Tabular Solution Methods

In RL, tabular methods [6] refer to solutions that represent the value function or policy using tables or arrays to store values of state-action pairs. Tabular methods are typical for small and moderate-sized state spaces where the table size remains feasible.

Sutton et al. [6] use the multi-armed bandit (MAB) problem to establish essential terms and principles. MAB, also denoted as k-armed bandit problem, formulates a learning problem where the agent faces a repeating decision-making problem between k options or actions. Each of these actions has a value based on an expected or mean reward. An action selected at time step t is denoted as A_t while the corresponding reward is denoted as R_t. The value of an action is denoted as $q(a)$, so that the expected reward for an action a would be:

$$q(a) = \mathbb{E}[R_t | A_t = a],$$

for which the estimated value $Q_t(a)$ would optimally be close to $q(a)$. In the notation, v, a, s and q denote functions, states or actions, whereas V, A, S and Q denote a set values or estimates of functions. When the action value estimates are maintained, there is at least one action with the greatest value for each time step. These are greedy actions and the selection of greedy actions is called *exploitation* of the policy or current knowledge of the system. However, exploiting the knowledge excludes accumulating knowledge through *exploration*, so the state-action space must be thoroughly explored to ensure that the exploited policy is optimal for the particular problem. A thorough exploration ensures that the agent has the knowledge to select actions that may have a diminished imminent pay-off but end up with a better result when compared to a simpler, greedy policy. Exploration is usually enforced with the epsilon-greedy formulation in a discrete environment, where random actions are selected in a decreasing manner during the learning phase. In a continuous environment, exploration is ensured by adding noise to the actions or, if using neural networks, the network parameters [38].

2.5.1 Finite Markov Decision Processes

Markov Decision Processes (MDPs) [6] are a formalization of sequential decision-making. MDP consists of the agent, the learning decision-maker that interacts with the environment, which encompasses everything but the agent. Interactions are actions in the environment that alter the state of the environment from the agent's perspective. The environment also produces the reward for the agent, which the agent seeks to maximize w.r.t. its actions. Finite MDP is defined by a tuple that includes a finite set of states, a finite set of corresponding actions, a state transition probability function that gives the probability of transition from one state to the next w.r.t. given action and a reward function which represents the immediate reward when transitioning from current state to the next w.r.t. certain action. The set of states is denoted with S, while the action set is denoted A, the probability function is denoted as P and the reward function R, so that the finite MDP is defined by a tuple (S, A, P, R). The probability of a certain value of state-action pair at some time t is formulated as

$$p(s', r|s, a) = Pr\Big(S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\Big),$$

for all $s', s \in S, r \in R$ and $a \in A(s)$, defining the transition dynamics of the MDP. If the state encompasses all the information of the agent's interaction with the environment that has an impact on the future, it is said to have *Markov property*. Therefore, assuming Markov property, knowledge is required solely on the present state and action to decide for the next action, independent from the history beyond the present state. Several useful entities can be computed from the four features of MDP: state-transition probabilities, expected reward for state-action pairs and expected rewards for state-action-reward-next state-action sequences.

The MDP is an abstract framework suitable for a multitude of problems, as it is an abstraction of goal-directed learning from interaction. It reduces a goal-directed learning problem to consist of two to four aforementioned arguments and provides a useful framework to formulate reinforcement learning problems.

2.5.2 Dynamic Programming

Dynamic Programming (DP) methods [6] were introduced by Richard Bellman [39] and use *Bellman equations* to find optimal policies. Policy iteration and value iteration are the most well-known and established DP algorithms. It is generally assumed that the environment is a finite MDP so that the sets of MDP are finite and dynamics are given by a set of probabilities $p(s',\ r|s,\ a)$ for all $s \in S$, $a \in A(s)$, $r \in (R)$ and $s' \in S^+$ where S^+ includes the terminal state or states if the problem is episodic. The idea of DP is to find optimal policies using value functions v or q which satisfy the Bellman equations [6] displayed in Equations 2 and 3:

$$\begin{aligned} v(s) &= \max_a \mathbb{E}[R_{t+1} + \gamma\, v(S_{t+1})|S_t = s,\ A_t = a] \\ &= \max_a \sum_{s',r} p(s', r|s, a)\Big[r + \gamma\, v(s')\Big] \end{aligned} \tag{2}$$

or

$$q(s,a) = \mathbb{E}[R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') | S_t = s, A_t = a] \\ = \sum_{s',r'} p(s', r|s, a) \Big[r + \gamma \max_{a'} q(s', a') \Big], \tag{3}$$

for all $s \in S$, $a \in A(s)$ and $s' \in S^+$. DP algorithms use these equations as update rules to improve the approximation of the value functions. In evaluating the state-value function v_π for an arbitrary policy π, known as *policy evaluation* in DP, the value function [6] is formulated as

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \Big[r + \gamma v_\pi(s') \Big],$$

which can be utilized as an update rule by exchanging $v_\pi(s)$ into $v_{k+1}(s)$, known as the iterative policy iteration algorithm. In essence, the policy iteration evaluates the value of the current policy as the sum of the products of the probabilities of actions and the cumulative reward with a discounted value of the policy estimate for the next state w.r.t. actions taken.

For value iteration, the policy evaluation steps can be formulated as

$$v_{k+1}(s) = \max_a \sum_{s',r} p(s', r|s, a) \Big[r + \gamma v_k(s') \Big],$$

for all $s \in S$, which is the Bellman optimality equation turned into an update rule where k denotes the successive approximations after arbitrary initial approximation v_0. Compared to policy iteration, value iteration can stop earlier than policy iteration, where the convergence occurs only in the limit. The update of estimates of the state values based on the estimated values of successor states is called *bootstrapping*, apparently derived from the phrase *to pull oneself up by one's bootstrap* [40]. In this context, bootstrapping refers to updating estimates based on other estimates, thus pulling oneself up by one's bootstrap self-sufficiently.

The problems related to DP methods relate to the requirement of iterating through the whole state set of the MDP, which becomes infeasible given a very large state set.

2.5.3 Monte Carlo Methods

Monte Carlo (MC) methods [6] simulate episodes of interaction with the environment and estimate value functions to learn optimal policies by averaging the returns of these interactions. Characteristically, Monte Carlo methods require solely samples from the interactions with the environment, regarded as *experience*, without complete knowledge of the environment.

MC methods require [6] complete episodes, i.e. from start to termination, to estimate values and update the policy. MC methods can be used to estimate either state values $v_\pi(s)$ or state-action values $q_\pi(s, a)$, if a model of the environment is not available. The exploration of states is initially ensured by giving all states a non-zero

probability to be selected so that the policy considers as wide a range of possible states and actions as possible.

2.5.4 Temporal-Difference Learning

Temporal Difference (TD) [41] is a learning procedure for prediction problems which is driven by the difference between temporally successive predictions. Instead of comparing the prediction to the actual outcome, the TD approach compares the previous and current predictions. $TD(\lambda)$ family of learning procedures denotes the weight increment based on the observations w.r.t. their recentness denoted by λ. Thus, a common procedure TD(0) denotes performing a weight increment determined only by the effects of the prediction done with the most recent observation.

A simple TD(0) method updates the estimate $V(S_t)$, where V denotes the array estimates of the set of all nonterminal states S at time t, immediately on transition as

$$V(S_t) \leftarrow V(S_t) + \alpha\big[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\big],$$

when receiving the reward R_{t+1}. Compared [6] to the previously mentioned Monte Carlo updates with the target $G_t \leftarrow \gamma G + R_{t+1}$ the TD(0) update target is the $R_{t+1} + \gamma V(S_{t+1})$. The TD target estimate is, in essence, a combination of the sampling in Monte Carlo with the bootstrapping of DP. This results in several advantages over MC and DP methods. TD does not require a model of the environment nor complete episodes, which makes it straightforward to implement.

The TD methods can be expanded to *n-step bootstrapping* [6] as a unification of the one-step TD methods and MC methods, enabling smooth transfer from one method to the other according to the demands and features of a particular task at hand. For n-step bootstrapping, the target for arbitrary n-step update is formulated as

$$G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(s_{t+n}),$$

as an approximation of the full return where the missing terms are corrected by $v_{t+n-1}(s_{t+n})$ or taken as zero if the n-steps go to or beyond episode termination.

SARSA (State-Action-Reward-State-Action) algorithm is an on-policy TD control algorithm that estimates $Q \approx q_*$ by choosing an action a based on current state s with an ϵ-greedy policy. When action a is taken, reward R and new state s' are observed and following action a' is then selected similarly while updating the policy as $Q(s,a) \leftarrow Q(s,a) + \alpha[R + \gamma Q(s',a') - Q(s,a)]$ and $s \leftarrow s'$, $a \leftarrow a'$ until terminal s is reached, $\alpha \in (0,1]$ denoting the step size.

Sutton et al. [6] state that one of the early breakthroughs in RL was the off-policy TD control algorithm labelled *Q-learning*, introduced by Watkins et al. [42, 43], which is defined as Equation 4:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\big[R_{t+1} + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)\big], \tag{4}$$

where the action-value function Q approximates the optimal action-value function q_* independent from the followed policy. This stable in RL has lent the name for Q

functions when regarding action-value functions, and it enabled early convergence proofs and simplified analysis, and $Q - learning$ has been guaranteed to converge to optimal policy q_* under the assumption that all state-action pairs continue to be updated during visits determined by the followed policy.

2.5.5 Q-learning

One of the major advancements in RL is the aforementioned *Q-learning*, a model-free algorithm introduced by Watkins [42]. The learning of the agent proceeds similarly to TD in a Markovian domain, as the agent tries out actions and evaluates the consequences as rewards or penalties received while estimating the value of the state to which it proceeds. The optimal action-value function is approximated with a Q function [6] that learns the values as displayed in Equation 4, so that the Q value is essentially the expected discounted reward for an action a_t in the state s_t leading to succeeding state s_{t+1}, while γ denotes the discount factor and α denotes the learning factor, a positive value between $]0, 1]$, as the value 0 would disregard the update. The algorithm evaluates the value of the state as the discounted sum of the succeeding states and their estimated values w.r.t. the probabilities each of the succeeding states is proceeded to.

Watkins shows Q-learning converges [42] into an optimal policy as $n \rightarrow \infty$ in any finite Markov decision process by maximizing the expected value of total reward over all successive steps from the current state. The convergence guarantee holds under the conditions that there is an infinite number of observations of the form $[s, a, r, s']$ denoting the current state, action, reward and succeeding state, as well as a positive monotonically decreasing learning factor α and that the sum of learning factors for observations is infinite.

2.6 Approximate Solution Methods

As the state space of problems can be arbitrarily large [6], the tabular methods can be extended to approximate solution methods to find a good approximate solution instead of going through the vast state space to find an optimal policy or value function. In principle, approximation methods aim to generalize over the explored states and actions to be able to make good decisions in previously unseen new states. The approximation is performed with a parameterized function, such as a neural network, in which case the solution is referred to as deep reinforcement learning as it combines the two paradigms. While function approximation is an approach in supervised learning, reinforcement learning brings about issues such as nonstationarity and delayed targets. The nature of RL differs from conventional supervised learning, which approximates a function to map data point features to target values, but RL can utilize similar approximation tools such as stochastic gradient descent.

One advantage of approximate solution methods is the fact that they function in partially observable environments due to the generalization to unseen data [6] and thus showcase as the go-to methodology for the solution in this thesis.

Approximation can be divided into prediction and control, in which prediction methods approximate state to value $S_t \rightarrow U_t$ to evaluate the performance of a given

policy or value function and, through updates, to maximize cumulative rewards while control methods use state-action pairs $S_t, A_t \rightarrow U_t$ to select optimal actions to maximize the expected cumulative reward over time. Approximation methods are discussed briefly in the following subsections.

2.6.1 On-policy Prediction with Approximation

In approximation [6], as a contrast to tabular methods, the value function is not a table but a parameterized function with a weight vector $w \in \mathbb{R}^d$. A general stochastic gradient decent method for state-value prediction can be written as

$$w_{t+1} = w_t + \alpha[U_t - \hat{v}(S_t, w)]\nabla\hat{v}(S_t, w),$$

where α is the learning rate, $\nabla\hat{v}(S_t, w)$ the vector of partial derivatives w.r.t. components of the vector w and U_t is an approximation of the true value $v_\pi(S_t)$.

On-policy refers to agent learning by improving the current policy. For prediction, the objective can be defined as

$$\bar{VE}(w) = \sum_{s \in S} \mu(s)\big[w_\pi(s) - \hat{v}(s, w)\big]^2,$$

which is the mean squared value error giving a measure of how much the approximate values differ from the true values.

Example algorithms for on-policy learning include the previously mentioned SARSA and, for function approximation, Trust Region Policy Optimization (TRPO) [44] and Proximal Policy Optimization (PPO) [45].

2.6.2 On-policy Control with Approximation

The general gradient-descent update in on-policy control [6] is

$$w_{t+1} = w_t + \alpha\big[U_t - \hat{q}(S_t, A_t, w_t)\big]\nabla\hat{q}(S_t, A_t, w_t),$$

that accounts for past actions to enhance decision making instead of mere state value prediction. The U_t update target denotes any approximation of $q_\pi(S_t, A_t)$, so it can be substituted with, e.g. Monte Carlo return (G_t).

For continuing tasks, which have no definite termination state nor start state, the cumulative reward is different from the Bellman equation. While useful for tabular methods, the discounted reward does not function properly in an environment where there is no clear start and end and can be deprecated as it has no impact on the ranking (ordering) of a decision. Discounted returns are proportional to the average reward [6], which regards delayed rewards with equal importance to immediate rewards.

2.6.3 Off-policy Methods with Approximation

Off-policy methods differ from on-policy in the way that the data is collected using a previous policy that interacts with the environment, while a target policy is the one

being improved. The target is *off-policy* as it does not dictate the behaviour in the environment. The behavioural policy is still usually updated on-policy, after which the policy results are compared to those of the off-policy target. The target policy is then updated according to the learned approximations achieved with the behavioural policy.

Example algorithms of off-policy RL are Q-learning [6] and its function approximation extension Deep Q-Networks (DQN) [46], and for continuous action spaces Deep Deterministic Policy Gradients (DDPG) [47].

2.6.4 Policy Gradient Methods

As a clear distinction to the former approximation methods, policy gradient methods [6] aim to learn a parameterized policy that can select actions without knowledge of their value. The learning phase may utilize the value function to learn the optimal policy parameters, but the policy functions in action selection without it. In some cases, learning the policy may be a simpler function to approximate. The policy updates smoothly in comparison with ϵ-greedy selection, can inject prior knowledge of the policy into the RL system and has a natural way of finding stochastic optimal policies in environments that exhibit an action selection of arbitrary probabilities.

2.7 Multi-Agent Reinforcement Learning

As described by Albrecht et al. [48], *multi-agent reinforcement learning* (MARL) is based on reinforcement learning, but instead of optimizing a single agent to make the optimal decisions in a particular environment, MARL focuses on learning optimal policies for multiple agents.

In the environment, the agent's objectives can be competitive, cooperative or mixed [48, 49]. Fully cooperative means that the agents' objectives are aligned so that collaboration is required to achieve the desired, shared goal. In a competitive scenario, the agents' objectives may be opposed so that the agents are in direct competition with each other. In the case of this thesis, the environment consists of cooperating agents, meaning the agents on one side are in a competitive environment against the agents on the other side. Hence, the use case is mixed MARL. From the Blue force perspective, the MARL environment is pictured in Figure 3.

In Figure 3, from the Blue force perspective, the opposing agents belong to the environment, while they operate within the environment precisely in the same manner as the Blue force agents. Another MARL aspect is the sharing or communication of observations. In this case, the agents share certain information immediately, such as the status of friendly units and the number of spotted opponents. Implementation-wise, when using deep RL, the designer can opt to have separate neural networks for each agent or a common network for all.

There are several challenges in MARL. The fact that there are several learning agents causes non-stationarity [48], as not just the environmental aspects keep changing but the policies of agents as well. This may lead to cyclic or unstable learning dynamics.

Another issue is the difficulty of defining an optimal solution [48]. In a single-agent system, the optimality of a particular policy is easier to define, as it should yield

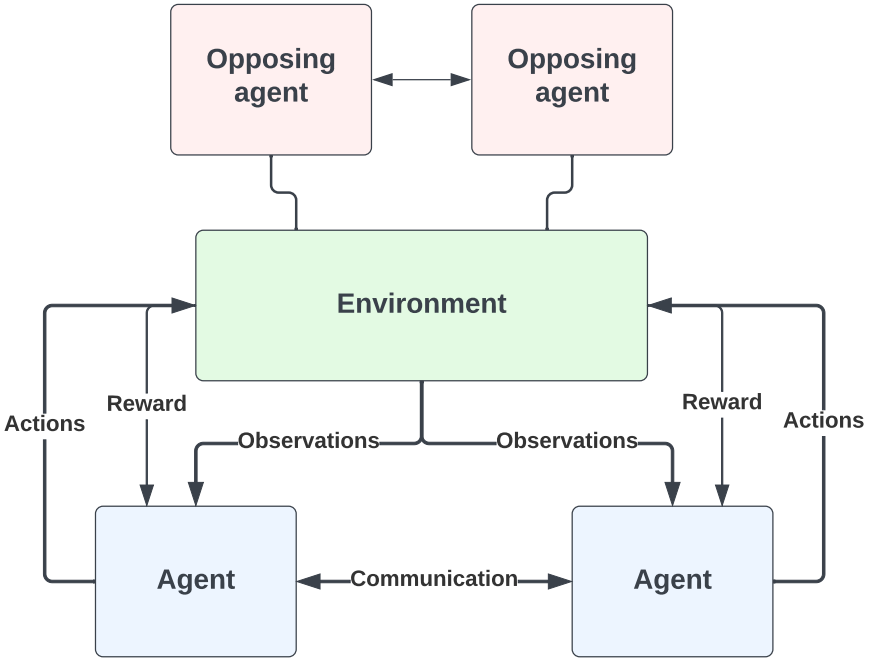


Figure 3: Multi-Agent Reinforcement Learning framework [48]

maximum expected returns for each state. In MARL, the goals of each agent may be very different.

The third challenge is rewarding correct agents [48], i.e. assigning credits when due, for agents' participation in some action. In this littoral combat scenario, a ship might engage a vessel that has been targeted by another unit, and the reward should be assigned accordingly for the successful chain of actions.

The last issue named by Albrecht et al. [48] is the scaling in the number of agents, as in the case of this thesis. This is due to, for example, the architecture and parameter selection of neural networks in deep RL.

The solutions for said MARL challenges in this thesis are described in sections 4 and 6. MARL is discussed in detail in, e.g. MIT MARL book by Albrecht et al. [48].

2.7.1 MARL environment assumptions

As many design choices and assumptions exist regarding MARL, the preliminary approach to formulating littoral naval warfare is discussed briefly.

As previously stated, the game environment must be partially observable to reflect a real-world warfare scenario. The units cannot continuously monitor the whole battle space. The units communicate to form a common operational picture used by the higher echelons and the units themselves. Only unit status and targeting data have to be shared between cooperating units, to enable cooperation. As such, the environment is designed to be partially observable with joint observations. Each contributing unit observes its surroundings to decide its manoeuvres in the vicinity, shares information about its location and status, and targeted opposing units with other agents.

Another aspect is the execution of agent actions. To avoid favouring either side in

the environment, the observations are gathered at a time step t, resulting in common albeit partially observed s_t for each unit, and the actions are derived from these as $a_t = \pi(s_t)$. As the actions, such as engagements, affect the environment, the actions need to be executed simultaneously.

The final assumption is that the agents are reactive, i.e. the decisions are based on current observations without memory or other guiding policies. This approach is selected mainly due to simplicity, not only in implementation but due to the difficulty of designing a suitable mechanism without enforcing a known tactical solution. This bias cannot be fully avoided, as it persists in all design choices, from the perception of observation space to environment features.

The memory mechanism could be a recurrent neural network solution or an attention mechanism. These are not included at this point and are left to be researched in future studies. Other solutions to use non-reactive agents include, e.g. end-to-end learning, such as displayed by AlphaStar [9] that map the raw inputs into the decision. This could be done by processing the current situational picture to the network as an image. Additionally, a model-based policy or a hybrid solution could be used. The design choice of reactive policy is discussed later with results from the experiments.

3 Naval Warfare

This section examines naval and littoral warfare to establish the theoretical framework and produce a knowledge base for problem formulation. The section briefly introduces the basic principles of naval warfare spanning from the strategical level into the tactical level to provide grounds for feature selection and modelling in the developed environment.

3.1 Strategic principles of naval warfare

Captain Alfred Thayer Mahan [50], a recognized maritime strategist and historian, defines the separation between strategy and tactic as *contact*, the point at which opposing sides meet and engage each other. The sides have been brought to the tactical theatre by strategy, either to meet the opponent to deny its strategic objective or beforehand to ensure the location for own means. Mahan [50] says that the sea poses its worth from a political and social point of view as a great highway, and the necessity of a navy arises from the need to ensure commercial shipping, upon which countries with access to the sea are heavily dependent.

Sir Julian Corbett [51] states that the object of naval warfare "must always be directly or indirectly either to secure the command of the sea or to prevent the enemy from securing it". He also notes that the sea's common state is uncommanded; no side has command over it. As pointed out by Mahan, the strategic evaluation of the sea area's importance results in directing forces to assume command over it or to prevent the opposing side from doing so. The idea of the strategic object aimed at by using naval warfare differs between Mahan and Corbett, where Mahan imposes the idea of destroying the opponent's sea power to ensure their command of the sea and Corbett focuses on using the naval power to disrupt the commerce and use of the sea as the means of transport to bring the opponent to yield, in a constricted sea area these objectives merge into the same encounter where the battle for the command of the sea area requires the contact between opposing fleets in order to reach the objective of command of the sea and continuation of sea commerce.

Consequently, it can be said that in the scope of this thesis, the underlying strategic interest that results in the research problem is that of assuming control over a sea area. The control is exhibited with the use of naval units in order to assume said control through gathering recognized maritime picture to support the naval operation and engaging opposing units to enforce and maintain control of the area while the other side aims to deny this objective. In the context of the Finnish Navy, which is tasked with securing sea communication and repelling naval attacks [52], the thesis focuses on the northern Baltic Sea.

Finnish Navy officer, researcher and strategist, commander Vänskä [53], has written that despite a striking evolution in appearance, the basic principles that shape naval warfare have remained unchanged through centuries. Vänskä points out that the *young school of thought* in naval warfare for small navies focuses on having an effect on sea lines of communication, denial of control and coastal defence. He also shares the view of Corbett [51] that naval warfare is always tied to land warfare, as purely naval

warfare does not exist, i.e. naval warfare serves the goals that are realized on land.

In summary, the strategic principles of naval warfare can be derived as

- The sea lines of communication pose value to every nation with access to the sea
- Sea is under one's control solely in a situation where the control is exhibited through sufficient force projection
- Control of the sea is usually limited in time and space
- While one side aims to seize control of an area, the opposing side may aim solely to deny this objective
- War at sea serves the strategic aim at some other level or location

In short, naval warfare aims to preserve the use of sea area for commercial and military purposes or prevent the opponent from using it accordingly. The goal is reached through force projection to serve the established objective. On a global scale, oceans have played a major part in many modern conflicts since the beginning of the 20th century. In a littoral context, these large-scale objectives and purposes hold, but in a smaller scale and more intense tempo, as described in the following subsection.

This thesis focuses on the tactical aspect, as defined by Mahan [50], not considering the strategic aspects and decision-making that lead to contact between opposing forces. Instead, in order to assess using RL for decision-making support on a tactical level, the aforementioned strategic principles are considered to have led to the imminent engagement between parties, and the focus of the study is to optimize the encounter for both sides to test the hypothesis that a working RL solution either mimics the current tactical ideas of littoral naval warfare or proposes new solutions for unit deployment.

3.2 Littoral warfare

Littoral warfare is globally scarce, as most navies operate in open, non-confined oceans. As this thesis is focused on the littoral environment, the environment requires a background, which is established utilizing the principles described by Milan Vego [54], professor of joint military operations in U.S. Naval War College, who is a widely credited author in naval warfare with a deep insight into constrained sea areas such as the Baltic Sea. The term "littoral" refers to a "coastal region" or a "shore" [55]. The features described by Vego are also echoed by Finnish Navy officer and researcher Vänskä [53].

Vego [54] notes that littoral naval warfare differs from warfare in the open sea due to the complex, dynamic and challenging physical environment, offering challenges and opportunities for naval forces. While objectives are regarded as similar to those in the open sea, the way to accomplish the objectives differs. It is said that littorals require decentralized command and control to perform well in a rapidly developing, enclosed environment. Vego notes that land-based aircraft and conventional submarines are

primary capabilities in the littorals along with multipurpose corvettes, small fast attack crafts and unmanned units. The most serious threat to large surface vessels is posed by anti-ship cruise missiles (ASCM). Due to this and the fact that surface warfare is one of two main warfare categories in the Finnish navy, the game environment for this thesis focuses on ASCM engagements. For the terminological consistency of the rest of the thesis, the ASCM will be denoted as SSM, an abbreviation for a surface-to-surface missile.

In an enclosed sea area, the distances between opposing sides are short and allow for deployment-replenishment-redeployment cycles unfamiliar to vast distances acquainted with naval warfare in the open oceans. [54] Despite the narrow sea allowing, or favouring, the use of all types of aircraft, it was chosen to focus on surface warfare executed by naval surface units. However, the built environment allows flying units to be adapted for reconnaissance and engagement missions. Similarly, subsurface units can also be included in the environment but are not in the scope of this thesis.

Due to physical characteristics, the littoral area limits the flow of traffic and use of the sea area. Characteristically to the Baltic Sea, the archipelago imposes limitations but also provides cover. While small islands and shallows limit the traffic to certain routes, the topography provides shelter and makes identification of choke points easier. Vego notes [54] that Finland's coast is fronted by some 790 islands larger than one square kilometre, supplemented by some 178,500 islets, while the Swedish coast has about 98,370 islands and islets. Likewise, it is noted that the Baltic Sea is shallow, as most littoral and narrow sea areas are, restricting the employment of major surface combatants. Vänskä [53] declares that large vessels face heightened risks when operating in the Baltic Sea when compared to oceans due to surveillance and discrepancy in their intended purpose and requirements of the constricted sea area. The regional characteristics are said to favour fast and small combatants over size. Shallow water enables the use of all types of mines further to modify the operational environment to one's benefit. Vego [54] also notes that in general, electronic sensors used close to the coast are prone to degradation due to various factors such as electromagnetic and atmospheric anomalies and the presence of a large landmass and multitude of electromagnetic sources. Using shipborne radars to detect and identify low-flying targets may be difficult in the presence of land clutter [56].

Vego [54] also notes that the small size of the area allows both the attacker and the defender to maintain constant observation over large parts of the area. However, due to the aforementioned conditions of clutter and anomalies, the warning and reaction time in the littorals is reduced compared to that in the open ocean. The small combatants can take advantage by concealing their actions within the archipelago and launching SSMs from cover to reduce further the reaction time and probability of detection on the opposing end. Vänskä [53] notes that while most of the Baltic Sea can be surveyed with relative ease, Finland's coastal surveillance capabilities do not reach the central or southern Baltic Sea, which instead requires either presence at sea, air or satellite surveillance and utilization of international cooperation.

As prerequisites for success, Vego [54] states that suitable and diverse platforms and close cooperation among friendly forces are key elements. Other key elements include robust command organization, well-developed theory and sound doctrine, but

in this thesis, these are tested as the outcome instead of a prerequisite. The doctrine for unit deployment is the outcome of the RL model and, as such, either validates or challenges the current doctrine of littoral combat in surface warfare. Instead, the use of MARL methodology induces cooperation among friendly forces, and the developed environment makes testing of diverse platforms feasible.

Regarding the tactical manoeuvres in confined seas, Vego defines [57] that an attack is a combination of tactical manoeuvres, i.e. movement of the ships such as patrolling and surveillance and fire conducted by single or several platforms to accomplish a minor tactical objective. A *naval strike*, defined as a sequential or simultaneous series of attacks by single or multiple platforms, is deemed the most important method of employing naval forces to accomplish tactical objectives. Vego states that in a narrow sea and especially in the archipelago, missile and torpedo-armed surface combatants can carry out strikes from an ambushing position against a stronger enemy force. The word *engagement* is interpreted to consist of a series of related strikes and attacks aimed to accomplish the principal tactical objective, such as resuming control of the sea area; Vego states that practically an engagement aims to destroy or neutralize the main part of the enemy force.

In the formulated littoral combat scenario, according to the aforementioned principles of naval warfare, the opposing (Red) side enters the area from the South in order to assume command of the sea communications to enforce its strategic goals and bring the Finnish nation to its terms on other fronts. The schematic Finnish navy (Blue side) operates in its own operational environment in the Archipelago Sea and northern Baltic to counter the presence of the opposing force by neutralizing its capability to enforce its objectives. As such, the scenario correlates with the strategic principles of naval warfare fitted to the Baltic Sea theatre's littoral tactics and enclosed environment. These are the determined objectives of the operation as per OR method formulation.

3.3 Military Decision Making Process

A decision-making process, in general, consists of observing some information and choosing the most suitable course of action to proceed. In the military, a common concept of a decision-making process is the *military decision-making process* or MDMP [58]. In the United States Army ADP (Army Doctrine Publication) 5-0 [58], it is stated that "the military decision-making process integrates the commander and staff in a series of steps to produce a plan or order". As this thesis aims to aid this particular decision-making process, it is discussed here briefly.

The MDMP [58] is described as an "iterative planning methodology to understand the situation and mission, develop a course of action (COA), and produce an operation plan or order". There are several steps that start from analysing the mission itself and proceeding to draft a set of possible COAs. These are analyzed and compared to come up with the best possible solution before promulgating the orders for implementation.

This thesis hypothesizes that RL will aid in COA development and analysis. The mission analysis should result in reward engineering if the RL environment is ready. The reward engineering should reflect the goal of the mission, after which the COA development could be executed with, hypothetically, MARL by optimizing available

resources w.r.t. opponent and its (presumed) objective.

The MARL could produce either one or several policies that represent different COAs with a partly ready analysis that would cover the results of testing it, i.e. how many losses are anticipated in each COA. This way the decision-maker's task could be enhanced with AI by producing data to support the otherwise quite subjective perception of the situation and its outcome.

Of course, on the other hand, there are the assumptions and bias that exists in the implementation that need to be considered alongside the results, but ideally, an RL or MARL solution could be able to either complement, challenge or augment the decision-making process in the COA development, analysis and comparison steps.

When brought into the concept of naval and littoral warfare, the same MDMP applies. The commander and the staff have to analyze the mission. As established above, on a strategic scale, the mission usually relates to the control of a sea area to either utilize it as a sea line of communication or to prevent the other side from using it as such. At the tactical level, the mission boils down to establishing a presence or engaging the opposing party with available assets in an optimal manner to ensure the best possible outcome. Next, the standard *kill chain* is introduced, which describes the process of an individual unit or a group of units in compiling a situational picture and awareness and acting on it, after which the surface combatants are briefly introduced to establish an idea of the concept.

3.3.1 Find, Fix, Track, Target, Engage, Assess

In all modern combat, there exists a *kill chain*, a process or model from finding a target to engaging the target. A common model for a kill chain is F2T2EA [59], which stands for *find, fix, track, target, engage, assess*. In this sense, to produce the results described in the principles of naval and littoral warfare, the situation is always similar to the naval assets when the mission or operation is in effect.

When assuming control of a sea area or denying the control of the opposing force, the naval units have to find targets to produce a recognized maritime picture. The recognized picture functions as a decision-making tool to reinforce the operation's goals, observe and monitor the ongoing situation and decide on which units to act on. The finding of a target is more vague than fixing; the fix is an exact location, whereas the finding can be an intelligence report or general whereabouts.

If the found target incites further actions, its movement needs to be tracked until a decision on follow-up actions is made. If the decision is to engage the target, it is targeted by assigning it to a weapon system. The engagement is performed with the assigned system, in this case, surface-to-surface missiles, to produce the desired effect.

After the kill chain has proceeded from the initial finding of the target to the engagement, the effect has to be assessed to determine the succeeding actions. If the engagement is successful, the kill chain has run its course, and the target has been neutralized. Otherwise, the chain starts over from a suitable step, depending on the status of the target.

3.3.2 Littoral Combatants

As described by Christiansen [60] in his thesis on the subject, large littoral combatants can be considered to be around 3,000 tons in displacement and four hundred feet ($\sim 120\ m$) in length while carrying 8 SSMs and boasting a top speed beyond 30 knots. Likewise, a small littoral combatant can be described, generally, as a vessel with a length of 200 feet ($\sim 60\ m$) and 500-ton displacement, with 4 SSMs onboard, similarly with a maximum speed above 30 knots.

These described features are echoed in the features of current Baltic Sea fleet complements, where small combatants are some 50-60m in length with approximately 4 SSM onboard, while large combatants are beyond 100m in length with a displacement several times larger and 8 SSM onboard. These features concur with Finnish PGGs, also known as Fast Attack Crafts or missile boats. The currently operational Rauma and Hamina class vessels [61, 62, 63] are $\sim$ 50 m in length, have a displacement between 200 and 300 tons and speed over 30 knots with SSM armament of 6 and 4 missiles respectively. Similarly, Finnish Hämeenmaa-class coastal minelayers [64] have a displacement of $\sim 1,300$ tons, are close to 80 m in length and have a maximum speed below 20 knots. These specifications resemble the German navy K130 corvette [65], and thus is deemed to resemble a *medium* sized littoral combatant. For a large combatant, the specifications are close to those described for the US Navy Littoral Combat Ship by Christiansen [60], as the dimensional specifications are also close to those of the upcoming Finnish Pohjanmaa-class multipurpose combatant [66, 67].

Thus, the combatants for the environment are classified as *small*, *medium* and *large* so that the small combatants are fast and pose a smaller RCS and mast height and missile capacity half the size of the large combatants, whereas medium combatants are similar to large combatants except for the speed, which is deemed lower than those of the small vessels and large vessels. This is due to *Froude number*, a ratio found by William Froude, who observed that ship models move with speed proportional to the square root of their length [68]. This is due to the waveforming, as a longer hull produces a longer wave which travels faster and thus resists the movement (speed) of the ship less than shorter waves. The only way to overcome this impact is by having a very light hull compared to its length. Therefore, the small combatants are able to reach higher speeds similar to those of the large vessels, but medium-sized vessels fall short in this ratio.

3.4 Problem Formulation

The real-world problem examined is the littoral naval encounter between two opposing parties, labelled *Blue* and *Red* in this thesis. The encounter results from conflicting strategic interests in using the contested area for own means, which are irrelevant to the problem solution. Instead, the solution focuses on the optimal execution of the encounter, i.e. developing an optimal course of action to optimize the use of one's resources to achieve one's own goals while nullifying the impact of the opposing side.

The physical realities comprise the topography of the littoral environment. These include short distances and sporadic coastal areas with islands and islets that inflict

and filter vessels' movement and affect electronic sensors' propagation. In addition, the engagements rely on using missiles, for which the probability of an outcome has to reflect reality to an acceptably realistic extent.

Essentially, the environment must include

- the area's topography
- the effectiveness and range of sensors in the area
- formulation for the defensive and offensive capabilities
- the probabilities for outcomes of an engagement
- the actions available for the surface units involved

The surface unit features must formulate their freedom of movement, speed, size, sensor range and weapon capacity.

After these features are implemented into the environment, MARL agents can be used to learn the problem solution. An optimal, or at least a working policy, represents a particular, simplified solution for a particular problem. Accumulating several differing policies accounts for alternative solutions for the decision-maker. The environment also has to provide quantitative information on each solution's performance for the decision-maker to analyze the strengths and weaknesses of each alternative course of action.

4 The Reinforcement Learning Environment

The previous Section 3 has established the basis for the solution formulation by mapping out the strategic and tactical objectives of naval and littoral warfare, the general principles of littoral tactics and the features of generic combatants. These include

- The strategic principles and goals that result in a littoral confrontation
- The tactical principles of littoral warfare, i.e.
  - Short distances and battle tempo
  - Topography and its utilization
  - The concept of engagements
  - The usual types of combatants
  - The principles of *kill chain*
- The impact of constraints in littoral environment, i.e.
  - Weather and atmospheric phenomena
  - The effect on the electromagnetic spectrum
- The military decision-making process
  - The focus of this thesis is on the course of action design and analysis

The framework, derived from game theory and decision-making, is a partially observable stochastic game (POSG) [5] that extends the Markov Decision Process [6] to multiple agents and introduces stochastic features in accordance with stochastic games [69]. As explained earlier, naval forces are composed of several units, thus favouring a POSG formulation, and the emulation of chaotic warfare insists on implementing uncertainties that cause stochastic features in the environment. An MDP can also be formulated as a partially observable MDP (POMD) and, in a multi-agent setting, expanded to multi-agent MDP, but with added uncertainties, POSG is a suitable framework for environment development.

A POSG between teams of players $b \in B$ and $c \in C$ is represented by a tuple $(S_i, A_i, O_i, R(s_1, s_2, \cdots, s_n), P_{a_i}, P_{e_i})$, where S_i is the finite set of states, A_i is the finite set of actions and O_i is the finite set of observations for players b_i and c_i respectively. P_{a_i} is the set of probabilities of transition to the next state while $R(s_1, s_2, \cdots, s_n)$ is the joint reward between players on one side. The POSG environment has n agents on two sides, and each agent executes a set of actions a_t according to joint partial observations s_t. The probabilities $p \in P_{a_i}$ to select particular actions are represented by the policy π, while the probabilities of environmental factors are described by probabilities P_{e_i}: P_{e_i} includes the probability of opponents' actions as part of the dynamic, stochastic environment. The state space consists of the environment grid with a finite number of

suitable states for each agent. However, the state space is expanded with stochastic features which are continuous. The actions are deterministic except for engagements which are impacted by uncertainty. The uncertainties are formulated using probability distributions. Other assumptions regarding the environment are discussed in Section 2.7.1; the agents are reactive as their decisions are based on local, joint observations of the current state of the environment without memory of previous states and have no model of the environment, for example.

To produce a game environment of the northern Baltic Sea region, a map of Baltic Sea physiography [70] was used as the starting point to create a grid that captures the topography of the area into two dimensions. The picture was modified to contain only essential information. At this point, the water depth and land area altitudes were neglected to keep the environment simple. It is noted that expanding the environment to encompass such features is straightforward and will be discussed further below. Figure 4 presents the image used to define the environment.

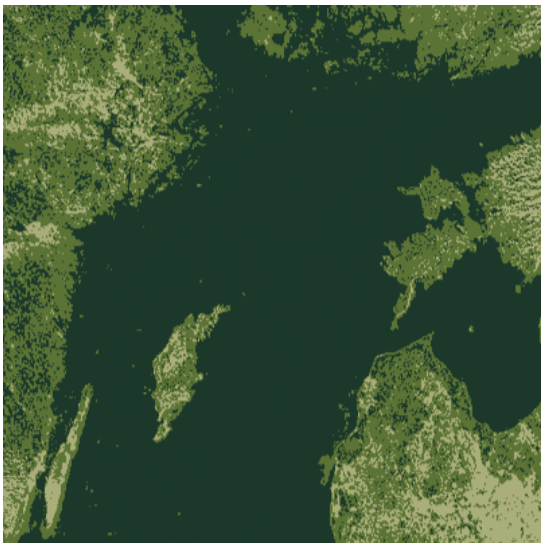


Figure 4: The modified physiography [70] image used to define the environment grid

The whole code used in this thesis can be found at https://github.com/valauri/Littoral-Naval-Warfare-MARL. The writing of the code has been sped up with GitHub Copilot (https://github.com/features/copilot/) and some methods have been initially implemented with ChatGPT available at https://openai.com/blog/chatgpt, before modification to fit the use case at hand. Using a method in the aforementioned code, the image in Figure 4 was reduced to the two-dimensional, 100 x 100 grid presented in Figure 5.

In Figure 5, the Baltic Sea area is pictured so that the open sea is dark, and the brighter it gets, the more elevated the terrain is. In this way, there is an option to declare certain thresholds for obstacle height that blocks, e.g., measuring electronic bearings or navigating over, depending on the vessel type. The 100×100 grid in which the units operate is drawn on the grey-scaled map. The red rectangle is the operational area of the Red force, as their side aims to seize control of the northern

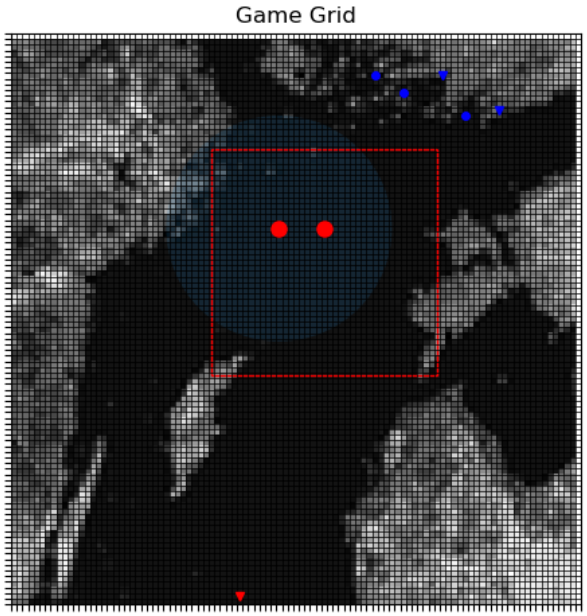


Figure 5: The resulting game grid with units and a visualized missile engagement

Baltic, as explained in the previous section. The Blue forces are pictured as blue dots and the Red forces as red dots, the size of the dot indicating the size of the vessel in question. The westernmost Red unit has its radar active, and the radar range is drawn as the transparent blue circle over the active unit. The triangles are replenishment points where units can replenish depleted missiles, colours indicating respective sides. Electronic bearings, meaning measured bearings of incoming opponent radar signals, are drawn as lines from the observer to the target and displayed in the colour of the observer's side. Missile engagements are also drawn as a line from the unit that fires to the measured target, marked with an $\times$, all coloured in orange. The engagement also displays a text that declares the number of missiles for the executed salvo. As a note, the visualization is implemented using Python Matplotlib [71] and is, thus far, extremely slow. The options to speed it up were investigated, but as the visualization does not inflict the research results, the method's slowness is acceptable.

The goal is to formulate a simplistic environment that exhibits features related to littoral naval warfare, such as relatively short distances and time spans. It can be said that an operation area of less than 200 nautical miles in width is a confined space for a navy vessel to operate in.

The primary function of a naval vessel is to transport a set of capabilities into a location where these capabilities are operated to produce the desired result, as described in the previous section. The most diverse set of actions for an intelligent agent relates to the movement of the ship, although it is essentially an auxiliary function to be able to execute the mission. As visualized in Figure 5, the archipelago is very schematic and thus does not exhibit major navigational challenges, unlike in the real world, where navigation in confined shallow waters requires expertise and practice. Nonetheless, the agents need to navigate themselves to an optimal position w.r.t.

upcoming engagements so that they can:

- Compile situational picture for targeting
- Engage targets when acquired
- Be able to avoid being targeted

In the conventional sense, the littoral scenario should favour, as Vego [54] puts it, the use of the Archipelago to remain hidden from enemy sensors and still be able to engage the enemy. This compromise of hiding behind obstructions and the mission of engaging the enemy in the open sea is the key contradiction the agents need to balance.

The mission for the Red side is to assume control of the Northern Baltic Sea by patrolling it with available assets. Hence, the Red units cannot utilize the protection of the topography in the Archipelago. On the contrary, the Blue side aims to deter the Red from the cover of the Archipelago and deny them control of the sea area. The next sections present the technicalities to exhibit these features.

As discussed in Section 2.7.1, the environment is partially observable even though cooperating agents share information to form a common operational picture. The common operational picture (COP) reflects the *common understanding of the current situation* and is usually incomplete. This design choice is elaborated in Section 4.1.2.

It was also stated that the agents need to make decisions at the same time. Therefore, each agent produces $a_t = \pi(s_t)$ based on the current situation observed by the agent, amplified with the observations shared from the other units on that current situation. After all the agents have inferred all actions for execution, the actions are executed. This may impact the execution of some actions, e.g. if a unit has decided to engage an opposing unit with 4 missiles and the target actions are executed in order before the unit's engagement, the target may move out of the designated missile target area, which foils the engagement. However, this reflects the reality of naval engagements. If the targeted location is not accurate enough to enable a successful engagement despite changes in the target location during the execution, the engagement would also be foiled in real life.

The agents react to the current, joint observations that comprise local observations amplified with relevant information from cooperating agents. As such, policies are reactive and past experiences of actions in the environment affect the neural network updates instead of the current state-to-action decision. In essence, the neural networks are trained to predict the action that yields the best outcome, reflecting a predictive or anticipatory policy. A different design choice, e.g., a recurrent neural network solution, would be memory-based and be able to utilize the past state-action interactions in the decision-making, making it memory-based instead of reactive. However, reactive policy is simple and fast and, as such, is a good starting point to solve the problem.

4.1 Environment features

This section identifies and formulates the features estimated as essential w.r.t. littoral naval combat scenario, as discussed before. Despite being described under the

environment, some features are agent-related and implemented as agent features, e.g., how the agent perceives the observations in the common environment.

4.1.1 Movement

Based on the satellite image, the game grid is approximately 500 kilometres times 500 kilometres in the physical world, making each grid cell a 5 kilometres x 5 kilometres area. Nautical miles (nm) are used in navigation, so each grid cell is approximately 2.7nm times 2.7nm. As discrete reinforcement learning algorithms rely on steps taken by agents, each step equals 15 minutes in the real world. Thus, a ship moving at a slow speed can move 1 square per step at the speed of 10.8 knots (nautical miles per hour). At medium speed, a ship can travel two squares, totalling 21.6 knots; at high speed, it can travel 32.4 knots. These are relatively standard operational speeds of navy vessels, although not all vessels are able to reach 30 knots.

As 15 minutes is a convenient and rather credible time frame for naval actions, the chosen game grid can be deemed to suit the needs as it is. The grid resolution should be increased to use more refined actions, for example, per minute. However, due to communication latency, the speed of naval vessels and the conduct of operations such as missile firing usually do not happen instantaneously; 15 minutes for each time step in the environment can be estimated to give an adequate representation of the real world the environment aims to simulate.

Therefore, the environment consists of a grid compassing of 100×100, approximately 2.7nm $\times$ 2.7nm squares, and state observations and actions represent 15-minute steps in the environment.

As the environment is grid-based and actions are initially discrete, the available movement actions include moving to any main compass direction as well as intercardinal directions, a distance between 1 to 3 squares per step (2.7 to 8.1 nautical miles), resulting in $3 \times 8 = 24$ available actions regarding movement for a ship that has the speed range from low to high (10.8 knots to 32.4 knots). The other option is to consider the actions as continuous, consisting of a direction and speed, which are then discretized to match a grid cell w.r.t. ship's current position. As 24 actions is quite a large action space to learn from, it was discovered while testing the environment that the continuous action space performs considerably better. Additionally, the 24 actions neglect all possible locations between the main and intercardinal directions and the option of not moving.

As the environment is created to serve purposes of the higher command, the navigational path-finding is done using A* algorithm [72] to verify whether a ship can sail to a grid cell selected by the agent. Otherwise, the navigational expertise to fill the higher command tasks is taken for granted.

The A* algorithm, as proposed by Hart et al. [72] in 1968, is a graph search algorithm that utilizes an evaluation function heuristic to determine the next favourable nodes to expand in the graph. The algorithm starts from a state s and calculates its value $f(s)$. Then it selects a node marked "open" according to the smallest value of f. If the node n is the target node, the algorithm terminates. Otherwise, n is marked closed, and f is calculated for each of its successors n_i, marking each child node

open if it is not already marked closed unless $f(n_i)$ is smaller than when it was last calculated for closing. Then, the algorithm repeats the steps by expanding the node with the smallest value of f. In the implementation, there are open and closed lists for the nodes, and the nodes have f as a value to check whether the node shall remain closed or be reopened. The heuristic f is the Euclidean distance between the start and target nodes.

Hence, to improve the idea of main and intercardinal movement options, the navigational feature is created so that the agent observes its surroundings as a 2D grid within the maximum speed range from its current position. With the maximum speed of 32.4 knots, this means a grid of $7 \cdot 7 = 49$ cells. The grid is used to display the available future positions for the vessel, whereas the direction and speed of movement rely on the current position and other observation features. When the new position is determined, the feasibility of navigating to that position is checked with an A* algorithm to see if the path is unobstructed. Some obstacles can be avoided as long as the detour does not deviate too much from the speed range available.

If the new position is reachable and not obstructed by terrain, the ship moves there simultaneously with other actions.

4.1.2 Tactical picture compilation and EMCON

With surface warfare in focus, the second feature that needs to be taken into consideration is tactical picture compilation and emission control. Tactical picture compilation, which covers the first steps of the common F2T2EA kill chain model described above, is reduced to consist of electronic intelligence and surveillance radars [73]. The latter means sensing the environment with surveillance radar, plotting targets and identifying those with ship measures such as correlation to recognized maritime pictures and intelligence reports. Using a surveillance radar is considered an active measure, as the ship compiling the surface situation picture emits actively to fulfil this purpose.

Emission control (EMCON) [74] is regarded as *"The selective and controlled use of electromagnetic, acoustic, or other emitters to optimize command and control capabilities while minimizing, for operations security ... detection by enemy sensors"* as well as avoiding mutual inference and/or avoiding enemy interference. Electromagnetic emissions include mainly radar and communications. Usually, the EMCON is enforced with different policies that dictate which emissions are allowed in each operation or operation phase. For the sake of simplicity, the environment is chosen to encompass only radar transmissions so that the ship's surveillance radars are either active or not. This can also be interpreted so that all emissions are *strangled* when EMCON is enforced, and all emissions are allowed when it is not. This is the chosen point of view because simulating communications between the ships would require an additional layer of simulation and, while feasible, is beyond the scope of this thesis. For a thorough simulation, the environment should encompass the communications between ships and the command with a certain number of options for communication and network optimization to determine if the ships are available for updates and commands and can share their observations of the environment. Likewise, the means of communication would require different representations in the electronic spectrum,

as low frequencies carry further than high frequencies, for example.

Electronic intelligence and targeting, in this case, is understood as passively sensing the surveillance radars of other vessels. Electromagnetic radiation includes radio frequencies and other emissions such as data links, but only radar transmissions are considered valid to keep the environment simple. This is incorporated into the environment so that if a vessel is transmitting with its radar, the bearing of the radar transmission can be spotted passively from a predetermined distance. If the radar is spotted from two or more vessels on the same side, a cross-fix, also known as a radio fix or mathematically the intersection of lines, can be determined to provide a target for an anti-ship missile launch. The inaccuracies in these measurements are taken into regard by adding Gaussian noise $\sim N(0, 1)$ to the observed bearings to make the targeting more realistic. The cross-fix is calculated as

$$x_{target}, y_{target} = \left\{ \begin{array}{l} x_{target} = \frac{\tan\alpha_1 \cdot x_1 \ - \tan\alpha_2 \cdot x_2 \ + y_2 \ - y_1}{\tan\alpha_1 - \tan\alpha_2} \\ y_{target} = \tan(x_{target} - x_1) + y_1 \end{array} \right\}, \tag{5}$$

where x_1, y_1, x_2, y_2 are the positions of the ships that have detected the signal, and α_1 and α_2 are respective bearings received. The nominator of x_{target} is the point-slope form of a non-vertical line $(m_1 \cdot x_1) + y_1$, where m_1 equals $tan(\alpha_1)$ as, by definition, the tangent function returns the slope of a coordinate w.r.t. horizontal axis. In this case, the environment does not use a particular coordinate system, and the horizontal orientation does not affect the outcome. If the coordinate system were north-oriented, the bearing would need to be tilted by adding $\frac{\pi}{2}$. In the environment, the grid starts in the upper-left corner, and x denotes rows while y denotes columns, i.e. the used system is inverse to the common (x, y) coordinate system. In the coordinate grid, as the length between the x_i and x_{target} times the tangent of the angle gives the length of addition for coordinate y, it follows that $y_{target} = y_1 + (x_{target} - x_1) \cdot \tan\alpha_1 = y_1 + \tan\alpha_1 \cdot x_{target} - \tan\alpha_1 \cdot x_1$ and similarly for x_2, y_2, which gives

$$y_1 + \tan\alpha_1 \cdot x_{target} - \tan\alpha_1 \cdot x_1 = y_2 + \tan\alpha_2 \cdot x_{target} - \tan\alpha_2 \cdot x_2$$

that can be written as

$$x_{target}(\tan\alpha_1 - \tan\alpha_2) = y_2 - y_1 + \tan\alpha_1 \ \cdot \ x_1 - \tan a_2 \cdot x_2,$$

which can be solved for x_{target} as displayed in Equation 5 and use the resulting value in turn to solve the point-slope form $y_{target} = \tan(x_{target} - x_1) + y_1$.

According to Payne [73], a radar detects a target if the return of the emitted signal is sufficient for the receiver. This value, measured in Watts, is equipment-dependent. The returning power P_r can be calculated as

$$P_r = \left[P_t G_x \times \frac{1}{4\pi R^2}\right] \times \left[\sigma \times \frac{1}{4\pi R^2}\right] \times A_e,$$

where P_t is the transmitter peak power, G is the beam gain, σ is the radar cross section that is explained below and A_e is antenna's aperture. In short, the equation calculates the emitted power over the sphere of $\frac{1}{4\pi R^2}$ times the radar cross section over a similar sphere times the antenna's aperture, i.e. the effective area that collects the received

energy. As the received power decreases with range and there is a threshold value for detection that denotes the sensitivity of the receiver, the maximum range can be calculated as

$$R_{\max} = \sqrt[4]{\frac{P_t G \sigma A_e}{(4\pi)^2 S_{\min}}},$$

where $S_{\min}$ is the minimum detected power for the equipment. However, as stated, this equation gives the maximum detection range given the known radar cross-section, output power, beam gain and antenna aperture, all independent of the terrain and horizon. The actual values are not the focus of this thesis due to their nature and the fact that such capabilities are usually classified. Payne [73] displays an example of an AN/SPS-49 radar, which gives a maximum detection range of 112km. This, however, only applies to targets within the radar horizon due to earth curvature and radar signal propagation. When the $K-$factor of refraction is assumed for normal atmosphere as $\frac{4}{3}$ [75], the radar horizon can be calculated as

$$R = \sqrt{2Kr} \times (\sqrt{h_1} + \sqrt{h_2}),$$

according to Ghasemi et al. [76], where h_1, h_2 are radar antenna height (mast heights) for the transmitter, and the target and r is the radius of the earth, approximately 6370km. As mast heights can be approximated as 15 meters for small vessels and approximately 30 meters for large vessels, the radar distance for detection would be $\sim 39km$ between a large and a small vessel and $\sim 45km$ between large vessels, equaling approximately 20 and 24 nautical miles respectively. Despite these mathematical models, most surface radars can extend their reach significantly further. For example, Furuno ship radars [77] have a monitor range scale from 0.625 to 96 nautical miles: with the above range equation, it would require the mast height of $\sim 465m$ for both masts to reach the maximum range the scale provides. The radar specifications do not state that the radar can detect and track a target at that range, but the range scale hints that the surface radar ranges are far greater than the formulation would allow to assume.

As explained, the presented theoretical radar ranges fall short of the equipment specifications and empirical evidence: navigation and surveillance radars with said mast heights are known to detect targets further, sometimes more than twice as far. Huang et al. [78] as well as Rautiainen et al. [79] have conducted empirical experiments on radar ducting, which occurs either due to evaporation or atmospheric phenomena. For evaporation ducting, the temperature difference plays a crucial role as the evaporating water forms an evaporation duct, which allows the radar signal to travel over the calculated radar horizon. Likewise, the atmospheric ducting increases the range over the horizon. According to Huang et al. [78] atmospheric ducting may have a stronger impact than evaporation ducting. As the phenomenon is not consistent, it is difficult to implement precisely. Therefore, it is taken into consideration by giving each episode a $k \sim 1.0 + Beta(1,3)$ value, which is used to scale the range to represent, in this case, randomly occurring conditions regarding atmospheric phenomena that affect the radar range. The Beta(1,3) distribution was selected to produce lower-end scalars for ducting rather than a uniform or tilted normal distribution, as this represents

the observations [78, 79] more accurately. It is also known that the weather impacts the use of a radar to a great extent, e.g. rain creates more reflective surfaces for the signal to scatter from, but weather is not separately formulated. Weather impact could be added when weather conditions, such as rain, are known to impact the planned operation, but simulating possible weather conditions was deemed unnecessary in this early stage of environmental development. As Payne [73] states regarding other sensors: "humidity, visibility, weather and so forth... are too variable to treat other than empirically".

Due to environmental restrictions and phenomena regarding the radar equipment specifics and range calculation, the modified radar range equation is used to determine if a target is within the range of the radar. As stated by Payne [73], "The size and ability of a target to reflect radar energy can be summarized into a single term ... known as the radar cross-section, which has units of m^2". The effect of radar cross section (RCS) in detection by radar is implemented as a fraction so that small vessels have a $RCS = 0.7$ that is used to decrease the detection range calculated with the above radar range equation. In contrast, large vessels with a larger RCS have a value of 1.0, so the vessels are visible as soon as they are within radar range. The implemented version is

$$r_{\text{radar}} = \sqrt{2Kr} \times (\sqrt{h_1} + \sqrt{h_2}) \times k \times \sigma, \tag{6}$$

where σ is the scalar representing RCS and k is the ducting factor which is solely a scalar $k \sim 1 + Beta(1, 3)$ to upscale the range to simulate different levels of ducting.

Electronic intelligence has a far greater range than radars, as the sensors need to detect only the incoming signal instead of the reflected, returning signal. For simplicity, the EW range is calculated by doubling the radar range calculated *without* the reduction induced by RCS, as the RCS has no effect on the visibility of a transmission from the measured target, and it is related solely to the reflectivity of the target. The EW range is calculated by doubling the radar range that has already taken ducting into consideration to reflect the current state of the atmospheric environment.

If a target is within radar range, the target quality is regarded automatically as precise enough for an engagement. The third option is to have a visual contact within the visible horizon, which also gives a solid target for an engagement. Within a short enough range, the engagement is performed with the main gun, which does not deplete missiles. These instances are estimated to be extremely rare in this thesis. No specific ammunition expenditure or hit probability calculations are implemented for main gun engagements.

To verify that the targets can be seen with a respective sensor from the observing ship, Bresenham's algorithm [80] is used to define a line between the two positions which is then checked w.r.t. obstacle threshold. Different thresholds are used for passive EW methods and active radar transmissions.

So far, the environment encompasses the topography, movement and EMCON, resulting in two action types for the agent: pathfinding within the environment and binary selection of transmitting with the radar or not.

4.1.3 Surface-to-surface engagements

Naval surface warfare or surface combat can be seen, in its modern form, to encompass all warfare that aims to destroy or neutralize enemy ships, usually by firing missiles or torpedoes. The platform used is not a determining factor, as anti-ship missiles can be launched from land, aircraft, submarines and surface ships. Likewise, torpedoes can be launched from aircraft, surface ships, submarines, and coastal stations. The scope of this thesis is in surface warfare between surface combatants, but the developed environment can also be extended to include air and subsurface capabilities.

As tactical picture compilation covers the first steps of the kill chain by finding, fixing and tracking the targets, only the engagement and assessment remain. To formulate surface-to-surface missile engagements, one needs to consider at least two uncertainties: the success rate of a missile to hit its target and the ability of the ship to counter the incoming missile. In his thesis, John Schulte [81] has examined historical pre-1994 SSM engagements in a littoral context. Due to the alignment of the context, this thesis, despite its age, is chosen as the source to derive the probabilities to produce the engagement logic.

In Schulte's thesis [81] he analyzes historical instances of SSM engagements in three categories: defenceless targets, defendable targets and defended targets. The defenceless targets are merchant ships and fishing vessels that do not possess anti-missile countermeasures. The defendable targets are warships that have been oblivious to the incoming missile strike and thus not taken any defensive manoeuvres. The third category consists of warships that have detected the incoming engagement and taken respective measures. Of the analyzed engagements against warships, approximately $\approx$ 34.5 per cent have been defendable but failed to react to the incoming engagement. The probability of a successful missile engagement in the post-1982 era is determined as 63 % against a warship that has not detected the engagement and 45 % against a warship that has detected the engagement and taken countermeasures.

A probability of detection is needed to formulate these probabilities into the engagements within the environment. From Schulte's data, a probability of 34.5 % can be derived, meaning there is a 34.5 % chance of not detecting the incoming engagement. The reasons for failing to detect an incoming missile may be human error, sensor malfunction, or enemy actions such as jamming. This thesis considers the single percentage to encompass all these variables for simplicity. The probability of detection should differ w.r.t. the EMCON status: if the target ship is using its sensors to survey the surroundings, it should have a higher probability of detection, i.e. the probability of going undetected should be lower. Likewise, it can be argued that the probability of detection should be lower for an engagement against a ship that imposes strict EMCON as it does not actively survey its surroundings to avoid targeting in the first place. This is formulated in the environment so that, against a target that is emitting with its sensors, the probability of the engagement being undetected is decreased to 24.5 %, whereas the probability of the engagement being detected is increased to 44.5 % if the ship is not transmitting with its surveillance sensors.

Therefore, a prior threshold exists for the posterior probability of successful engagement. If the engagement is detected, the probability of a hit is 45 %, otherwise

63 %. It can be argued, based on Schulte's thesis and, e.g., the sinking of HMS Sheffield in the Falkland War [82] and lately the sinking of Russian Federation cruiser Moskva in the war in Ukraine [83], that a single hit is enough to at least take the ship out of action, i.e. render it useless for combat before repairs, even though Moskva was hit with two missiles according to open sources it was evidently not only neutralized but sunk. Sufficiency of one hit can be regarded as a safe assumption for smaller combatants, as both aforementioned examples are considerably larger combatants, a destroyer and a cruiser, respectively. Therefore, the final probability of a critical hit can be formulated as

$$p(hit) = \left\{ \begin{array}{r} 1 - (1 - 0.63)^n \text{ if } r < p(d) \\ 1 - (1 - 0.45)^n \text{ otherwise} \end{array} \right\},$$

where n is the number of missiles in the engagement salvo, the probabilities are driven from Schulte's thesis [81], r is a random variable drawn from $U(0, 1)$ and $p(d)$ is the probability of detection and defined as

$$p(d) = \left\{ \begin{array}{r} 0.345 - 0.1 \text{ if ship transmitting} \\ 0.345 + 0.1 \text{ otherwise} \end{array} \right\},$$

where the arbitrary addition or subtraction of 0.1 is merely a selected number to formulate the increase or decrease in the ability to detect the incoming engagement if using surveillance radar. The actual, real-life value for such a value depends on a multitude of factors, including equipment, crew and procedures, as well as measuring these in some performance testing scenarios, which results in a strictly classified value that reflects military capabilities. Hence, in the scope of this thesis, a convenient 0.1 is selected to resemble the difference in capability without addressing the complexity of its determination. It is also of notice that a friendly unit could detect the incoming engagement for the unit in radio silence and relay the information without the silent unit going active. This, however, is not taken into consideration, as the communication between the ships is not implemented, and it would make the implementation more complex but arguably add very little value in this phase.

These simple equations give the probability of at least one hit for the fired salvo of missiles. It is of notice that Schulte's thesis [81] is from 1994, and the probabilities are thus obsolete, but as this thesis aims to present the use of MARL in formulating littoral naval warfare the actual, tested real-world probabilities of state-of-the-art SSM are not relevant and additionally heavily classified information. By selecting these probabilities as the basis for the environment, no existing SSM system is compromised. Further, if the implementation functions well, a similar one can be utilized to simulate combat scenarios with the classified performance insight.

Another option is to formulate the probability as the probability mass function of a binomial distribution to determine the probability of exactly k hits as

$$P(k; n; p) = \binom{n}{k} p^k (1 - p)^{n-k},$$

where k is the number of successes out of n trials and p is the success probability of each trial. The binomial distribution should be used if the estimated salvo size depends

on the target vessel, i.e. there is data or knowledge of a doctrine or procedure that states the size of a salvo against a certain ship type or class. Then, the probability of a hit can be calculated from the cumulative distribution function on a certain interval that has the accepted number of hits as

$$P(k : m; n; p) = \sum_{k=1}^{m} \binom{n}{k} p^k (1-p)^{n-k},$$

where m is the number of hits required for an optimal desired impact while k is the lower bound of hits in an estimated successful engagement. However, as stated above, in this thesis, one hit is regarded as enough to take a ship out of action and thus, the binomial cumulative distribution function is not implemented but rather the simplified calculation of at least one hit.

Hence, the engagement logic consists of determining whether the incoming missile salvo is detected and then calculating the probability of at least one hit. Then, the success is determined if a random draw from $U(0, 1)$ is less than the calculated probability.

The assessment of the engagement could be done with an aircraft or intelligence which would confirm the effect of the executed strike, but in this scenario, the assessment is based on continuing surveillance: if the ship is neutralized (sunk) it simply disappears, and the engagement is deemed successful. The written program automatically produces feedback on the engagement's success without adding complexity to the assessment. This differs greatly from reality, as the confirmation for a successful engagement is difficult to perform in a real-world combat setting, but this is neglected for the sake of simplicity.

Regarding missile range, modern SSM [84, 85, 86] exhibit ranges up to and beyond 300 kilometres. As the main limitation in the range is the targeting range of surface vessels and not necessarily the range of SSM, 300 kilometres is selected to reflect the modern-day SSM range. It equals the distance of 60 grid squares. Each found target is checked to be within range before engagements.

4.1.4 Replenishment

As briefly denoted in Section 3, one typical aspect of the littoral environment is the tempo due to short distances. This also encompasses the rate at which units can be replenished and replaced.

In a combat situation, SSMs are quickly depleted if the force compositions are somewhat equal in strength. Therefore, the replenishment of missiles is crucial to carry on with the engagement if the initial salvo is not decisive.

To implement this engage-replenish cycle in the environment, the replenishment points were decided on and implemented in arbitrary but suitable locations for both sides, as well as considered in the reward function to encourage the execution of replenishment actions. Essentially, the replenishment would occur when the unit has depleted its missiles and then proceeded to the replenishment point. Optionally,

the missiles could be replenished at sea if an auxiliary vessel transports them to the vicinity.

Despite being a characteristic feature of the littoral environment and being originally implemented, the experimentation proved that the agents were unable to learn such behaviour on top of the engagements themselves. This was due to the selected training episode length. It can be argued that the agent could be different for units that have depleted their missiles so that they only guide the unit to the closest available replenishment point with no regard to the combat, for example. However, despite being recognized as a prominent feature, the replenishment solutions were later abandoned from the scope of the experiments in this thesis.

4.1.5 Formulating combatants

The combatant class was implemented to include 3 ship types: small, medium and large. A small ship would represent a PGG (Patrol Gunboat, guided missile), similar to those of Finnish and Swedish navies. As features, PGGs are approximately 50m in length overall, have a displacement of approximately 300 tons and maximum speed in excess of 30 knots. Usually, PGGs are equipped with 4 surface-to-surface missiles, like the Finnish Hamina-class PGGs [87]. In the environment, small combatants have 4 SSMs and can travel 3 grid cells ($\tilde{8}$.1 nm) each time step. The radar cross section is determined as 0.7, which does not represent the actual radar cross section but a scalar ratio that inflicts the range of detection as described in Equation 6.

The medium combatant is a hypothetical combatant between the small and large, with a larger number of missiles but with lower speed due to increased displacement but moderate length, as described in chapter 3.2.2. As such, medium combatants have the speed of 2 grid cells ($\tilde{5}$.4nm) per time step, 8 SSMs and an RCS of 1.0.

Finally, the large combatants represent a large corvette that can carry 8 SSMs and reach 30+ knots in speed (3 grid cells) and, like the medium-sized ship, has an RCS of 1.0.

Another consideration is the mast height, which directly affects the sensor range and is implemented as a combatant feature. Another neglected attribute is the draught, which would affect the ship's ability to navigate in shallow waters. For now, this is neglected, as the implementation would affect only the units operating in the Archipelago. The size of the vessels could be taken into account when determining the required number of successful missile hits to neutralize the target, as larger vessels tend to be able to sustain more damage. It is deemed irrelevant for the sake of the thesis, as the data is either classified or conflicting on the survivability of navy vessels w.r.t. SSM hits. Beyond these physical aspects, other weapon systems and their usage, such as surface-to-air missiles and torpedoes, could be implemented in the future.

An additional class, landing ship, is also implemented because landing operations provide another tactical game scenario to solve with MARL. However, the testing of landing operations is not reported in this thesis.

4.1.6 Reward function and winning criteria

To enable RL, the environment requires a reward function which the agents seek to maximize. Several reward indicators and the end goal are determined regarding the aforementioned features.

The end goal is deemed as neutralizing all opposing units. The episode is terminated if the end goal is not reached within a set number of episode steps. Initially, 40 time steps equal a ten-hour time span in reality, which is used as the maximum length of a single episode.

For the reward function, it assumed that for good learning results, the agents need feedback on movement, proximity to the area of interest, locating enemy units, engagement success, and eventually being victorious. As such, the initial formulated reward function, Equation 7, is

$$r_i = \Big(\sum_{i=1}^{n} \text{target} \Big) \cdot 3, \tag{7}$$

which is the sum of the target count, i.e. the reward is composed of the number of detected opposing units.

To guide the learning process by rewarding proximity to a certain location in the grid map, a small distance-related addition

$$\frac{1}{\max(\sqrt{(x - x_p)^2 + (y - y_p)^2}, 1)}$$

can be added. The x, y is the ship position and x_p, y_p is the determined focal point, giving an additional reward if a focal point like the centre of an operation area is defined, with a maximum value of $\frac{1}{1} = 1$ and decreasing while the distance to the focal point itself increases. It was also modified to

$$\left[\frac{\max(\sqrt{(x - x_p)^2 + (y - y_p)^2}), 1)}{\sqrt{2Kr_e} \times (\sqrt{h_1} + \sqrt{h_2}) \times k \times \sigma} \right]^{-1},$$

which is the inverse of the distance ratio to the assumed centre of gravity of the opposing force and the surveillance range of the vessel, reduced to a tenth to scale the additional reward to range $r_a \in [0.05, 7.7]$. This was implemented to encourage the units to extend their surveillance to the opponent's operational area, i.e. to guide the learning process, but it proved unsuccessful in enhancing learning performance and was later neglected.

The reward function was also layered so that it has two options: *aggressive* and *defensive*. This was implemented to encourage more tolerance for losses in early training by following the aggressive reward function, which could then be swapped to a defensive version. The technical difference is that in an aggressive version, the loss of units does not affect the reward. Only losing the whole episode has an impact. If the defensive mode were selected, the units would receive a penalty of $r_i = max(r_i - 5 \cdot n_{lost}, 0)$.

The Red units were also penalized in defensive mode with $r_i = \max(r_i - 2, 0)$ if they were not in or left the defined operational area after the episode had proceeded for 14 time steps or more. This restriction was implemented to keep the Red units from navigating into the Archipelago. While such a tactic could have been feasible and surprising, it would have conflicted with the goal of assuming control of the central area.

Engagements are rewarded in accordance with Equation 8:

$$\left\{ \begin{array}{c} r_i + 10 \times s, \ if\, s(\text{engagement}) = True, \\ r_i = \max(r_i - 5, 0), \ if\, T \notin \emptyset \ and \ \neg e \end{array} \right\} \tag{8}$$

where s is the return of $s()$ that determines the success of functionality, in this case, the success of each missile engagement performed, T is the set of spotted targets and e is the action to engage. This implementation is contradicting the initial reward displayed in Equation 7, as the penalty diminishes the reward received from finding the opponents: either the penalty or the initial reward could be removed to simplify the reward function, but as the displayed version worked to some extent as a result of extensive experimentation it was left as it is.

Movement actions are rewarded according to Equation 9:

$$\left\{ \begin{array}{c} r_i = r_i + 1, \ if\, s(\text{movement}) \\ r_i = max(r_i - 0.5), \ if\, s(\text{movement}) \end{array} \right\}, \tag{9}$$

where $s()$ defines, in this case, the success of the movement.

In other words, successful movement actions are rewarded with +1 and failure to engage found targets is penalized with −2 unless the resulting reward falls negative. Negative rewards were also used, but the tests did not prove to be successful with such solutions. If negative rewards are implemented, the approach should penalize all undesired actions and give high rewards solely to the end goal, i.e. successful engagements or victory in this case, to keep the scheme consistent. Otherwise, alternating positive and negative rewards for positive and undesired actions are difficult to deconflict.

Equation 10 implements a shared additional reward for victory or loss.

$$\left\{ \begin{array}{c} r_i = 0, \ if\, \text{episode is lost} \\ r_i + 100, \ if\, \text{episode is won} \end{array} \right\}. \tag{10}$$

The rewards are thus $r \in [0, \mathbb{R}+]$, and the agents should be able to maximize the reward function by aiming to locate and engage opposing targets. Cooperating units receive a reward for a successful engagement if they have not participated in the missile firing as

$$r_i + n_{hit} \times 2,$$

which would be more explicit if the participation in the targeting was tracked, which it is non-trivial to implement fairly as cross-fixing requires several unit participation, but radar targeting requires only one. In the first implementation, these are not separated; thus, assigning the reward for tracking solely to the unit that holds the target

is impossible. However, in the implementation, the positions of units and their radar status are communicated to all other units, so in this sense, the agents may be able to learn which combinations of locations and sensor usage result in rewards.

4.2 Summary of environment features

In summary, the environment needs to provide the agent with the following information for each time step:

- The current location of the ship
- Immediate surroundings within speed range for navigation
- If the ship is emitting with its surveillance sensors or not
- How many missiles are left for engagements
- Are there any enemy sightings
- Status of friendly (cooperating) ships

Therefore, the observation space for each combatant is to consist of the ship's position, EMCON status and missile status. These are supplemented with a (flattened) grid from the ship's position that includes the terrain within reachable distance. All enemy sightings are also included as an integer to the observation space, effectively meaning the number of spotted enemy ships. The same ship-specific position, EMCON and missile states are also included for all other friendly units. This means that the observation space consists of $4 \cdot n + (s \cdot 2 + 1)^2 + 3$ variables, where n is the number of ships on its side and s is the ship's speed. The last 3 values are the number of enemy sightings, the distance to the closest replenishment point and the ducting factor of the environment, which affects the range of the sensors. Usually, the naval units do not *know* the ducting factor but instead can observe it live on their sensors. The purpose of the distance to the closest replenishment point is due to the fact that units can deplete their missiles quite fast, and Vego [54] denotes that quick resupply is a key element in littoral areas. However, as the episodes are short and the agents have difficulties in performing the combat actions, the importance of replenishment actions is diminished.

The resulting actions for the ships to perform the desired tactical manoeuvres and engagements are thus

- Movement i.e. to choose where to navigate next
- Alter EMCON state if needed
- Fire missiles

If implemented as a discrete action space, the actions are selected as $argmax$ from the neural network output. As such, the movement takes up the grid around the ship within a reachable range. That would mean $7 \cdot 7 = 49$ options for a fast ship. The EMCON state, selecting whether to emit or not, is two actions and missile firing would be at least two actions, making a discrete action space for a fast ship 53. However, the missile firing action space should be the size of the salvo: if no missiles are fired, the action value is 0. Otherwise, the number of missiles selected for the engagement. Hence, the discrete solution requires 5 options for missile engagement if the salvo size is $[0, 4]$.

If implemented as continuous, the action space size is 4, and all actions are continuous values $a_i \in [0, 1]$. The movement can be implemented as a direction and distance only, then discretized into a grid location. The discretization is implemented as

$$(\lfloor x \rceil, \lfloor y \rceil) = (x + \cos d \cdot r, y + \sin d \cdot r),$$

where d is direction in degrees and r is the range of movement. To fit the equation above, the direction, originally $d \in [0, 1]$, is transformed to radians as $2 \cdot \pi \cdot d$ and then to degrees, and range by multiplying r with the ship's speed. Likewise, the EMCON state was discretized as $\lfloor EMCON \rceil$ and missile engagement as $\lfloor s \cdot m \rceil$, where $s \in [0, 1]$ and m is the number of available missiles.

For some implementations of MARL the environment needs a global state. In this implementation, the global state is concatenated from the observed states of the individual units as $n \cdot (4 \cdot n + (s \cdot 2 + 1)^2 + 3)$. This global state is processed with a respective value function.

The program architecture is as Figure 6 depicts. In a conventional sense, the environment provides its namesake for the agents to manipulate the combatant objects. The states are the observation states of the combatants for each side. The states are inferred into actions that are passed through the environment step function to the combatants for execution, after which the resulting changes in the environment and received rewards are returned to the agent. The agents train the respective networks according to specified algorithms to optimize one side of the game in one learning session, after which the sides are switched.

4.3 Identified constraints and design limitations

Regarding the implemented environment, the simplicity poses several constraints and limitations. First of all, the game physics is not implemented except for minor effects of the terrain. As the terrain is displayed in grayscale, the navigation has to avoid grid cells that exceed a selected threshold in value, as these are regarded as land. Using a different image could divide the map into shallow and deep waters more effectively, as the threshold can be different for different ship types, simulating different draughts of vessels. Likewise, the terrain affects the line of sight for all electronic intelligence, radar and optical sensors but is implemented with simple fixed thresholds. Nothing inhibits from adopting a more complex terrain, which would include ground altitudes

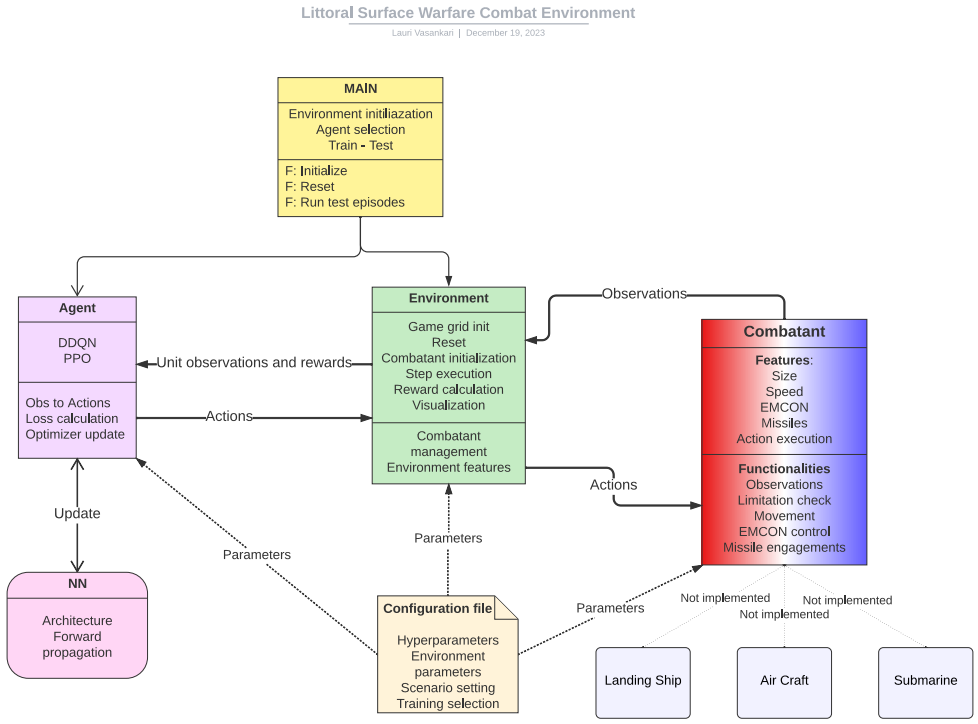


Figure 6: RL environment description

so that the mast height of the ships could be taken into account w.r.t. terrain when compiling tactical picture.

Another physical limitation is the design of the ships. The actual physical features such as RCS, which alters depending on the orientation of the ship w.r.t. detecting radar, are not taken into consideration. Further, a small RCS of a ship loses its edge when the distance to the target is shorter, but this is not taken into consideration either, as the orientation is not implemented. Both have an effect on the detection rate. Additionally, the close-in weapon systems and other countermeasures against anti-ship missiles are not implemented but rather regarded in the hit probabilities. Consequently, missile engagement planning is regarded as straightforward, as the orientation of the vessel has no impact on the ship's capability to counter the incoming missile salvo, unlike in the real world.

The use of a grid instead of a fully continuous game environment also imposes its constraints. The movements of the ships are abrupt instead of smooth, as the situation jumps ahead in 15-minute steps.

The design choices can be defended with the need for simplicity dictated by computational resources and the fact that the environment could be made very complex, in effect to the point of being a *digital twin* of the real world environment. Such an environment would require massive computational resources to be solved with MARL and may include too many features w.r.t. its purpose. This design approach aims to identify the absolute minimum of features implemented to produce a decision-making support system for a naval surface warfare squadron.

It is also noticeable that the opponent's real-world capabilities are usually unknown.

There is intelligence information on the capabilities and their performance, but real-world data is scarce and uncertain. Hence, the precise implementation of, e.g., RCS of ships and the effectiveness of close-in weapon systems can be seen as dubious.

A major limitation is the lack of a graphical user interface (GUI) with which a human player could play the game. This would enable the use of *imitation learning* or *inverse reinforcement learning* [88] to kickstart the learning process into the right path to find an optimal policy. Functionality with a similar goal is implemented for the Red forces by having the predetermined action sequence for movement and EMCON and opportunistic use of missile fire based on a probability that resembles the aggressiveness profile of the operation. However, the lack of GUI is a limitation that should be solved in the future. Currently, the environment can be visualized using Matplotlib [71] functionalities, which is slow and cumbersome. Using a game engine to produce the environment would also enable better options for the user to formulate the physical world and visualize the system. It would also allow the use of different observation space representations.

As the final point regarding design limitations, the environment does not include platforms other than surface combatants at this stage. Expanding the scope to multi-dimensional combat is feasible but will increase the complexity of the model as well as the presumed complexity of the algorithmic reinforcement learning solution beyond the scope of this thesis.

5 Testing of different RL algorithms

As per the previous section, the environment is quite complex even in its simplified form due to the large state space, which is affected by the partially non-static environment, alternating unit positions and probability-driven unit interactions such as engagements. Therefore, a tabular solution for the problem seemed infeasible. A tabular solution would suit a simpler environment, such as a deterministic scenario with a more limited grid size. Such an environment could be most likely solved with a Monte Carlo method by stochastically sampling different unit locations and results for engagements in each setting. This approach should yield a mapping of the most successful unit locations w.r.t. terrain for the selection of units to be victorious. However, this approach would be quite similar to the deterministic salvo method and its stochastic extension [89], upgrading almost solely with the impact of the terrain as the presented salvo model does not consider that.

A deep reinforcement learning setup was preferred for testing to utilise more modern RL methods. The complexity of the environment also encourages or even dictates the use of a neural network as the approximator. There are two possibilities: the discrete and continuous action spaces for the same environment. For the preliminary choice of algorithms, multi-agent DDQN (MADDQN) [90] was selected for the discrete environment and PPO in multi-agent setting for the continuous environment due to recent papers regarding Multi-Agent PPO (MAPPO) [25].

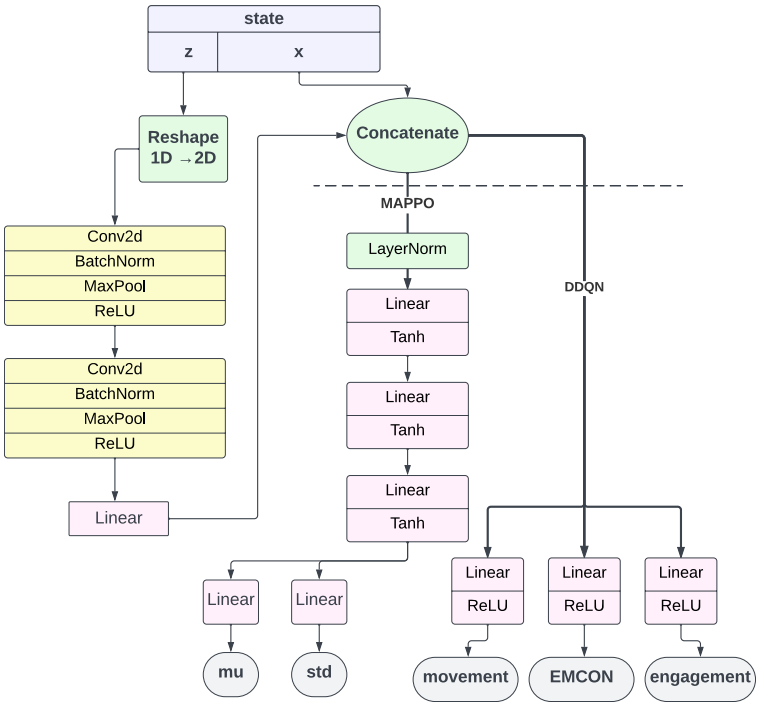


Figure 7: The schematic neural network

Figure 7 represents the neural network structure utilized. The input, i.e., the observation state, is passed to the network model, which slices and reshapes the

terrain observations into a 2D grid. This *image* is passed through a convolutional neural network structure whereas the unit states and other environment features are concatenated to the result of the convolution to be processed with linear layers and *tanh* i.e. hyperbolic tangent activation functions or rectified linear unit (ReLU) functions in the case of DDQN. In the end, the processed input is passed either to action-specific, separate linear heads in DDQN or to two linear heads that output the means and standard deviations for the normal distributions in MAPPO, which are used to sample the actions based on the observations.

The POSG framework was seen to favor DDQN solution as the experience replay it utilizes may help gather the scarce signals of the complex environment and learn from those, especially when the experience replay is prioritized. DDQN reduction of overestimation should also perform well with partial observations. Q-learning has been shown to outperform several RL methods in partially observable domains when combined with reward machines [91]. MAPPO was selected due to sample efficiency and suitability to continuous action space.

The MARL agents can be implemented either so that each unit is its own agent, having its own policy, or in a centralized manner that there is a common policy for all cooperating units. Because units can be neutralized and are therefore removed from the game along with their policies, the use of centralized training and policy was selected. *Federated learning* [92] could be utilized for decentralized training of units so that each agent learns a policy that is aggregated to a global model. This possibility remains to be researched in the future.

The device used for testing is a personal laptop with AMD Ryzen 9 5900HX CPU, 16BG RAM and NVidia Geforce RTX 3070 laptop GPU with 8BG memory and 5,120 CUDA cores.

5.1 Initial settings

Initially, the setup for training MARL agents was laid out as

- From 2 to 3 units supported for each side
- Fixed starting positions for each unit
- An initial "policy" was crafted for the Red force, i.e. predetermined actions
- The Blue side was trained first against the predetermined Red side
- The Red side was then optimized against the trained Blue side
- The other side was retrained against the new policy, if necessary

The number of units was limited to these options because it kept the testing feasible. All Blue units were small combatants in these tests, and all Red units were large combatants. Also, the design choice of using fixed starting positions would have required the manual input of starting positions for each vessel, so the number could not be arbitrary. Additionally, the value network likely requires modifications if the

number of units is very large or an alternative solution, such as a kernel, to project the individual states into a fixed-dimensional global state.

The design choice of using fixed starting positions resembles the intended use case: the units are in respective (known) positions, and the purpose is to develop courses of action from those positions. The opposing units are either in known positions due to intelligence information or a known area. This could be implemented by adding stochastic position selection, but it would further destabilize the already non-stationary environment and thus was not implemented.

In early testing, it became imminent that having both sides learning at the same time made the environment too unstable for the agents to converge on a working policy, as both sides were updating policies simultaneously, exhibiting the *non-stationarity* typical for MARL. The deviations in competing (and cooperating) agent policies came on top of the built uncertainties regarding action selection, engagement success, and alternating ducting factors. To stabilize the environment and enable faster convergence in early training, the Red side units were made to follow a preordained routine where the Red ships would commence their operation in the south and proceed to patrol the operation area with either 2 or 3 large (corvette-sized) combatants. For the preordained actions, movement and engagements were identical for each time step. The EMCON states were coordinated between the Red units so that the units performed tactical procedures of alternating emission spans between the units one after another, decreasing the probability of detection. The engagements were executed if targets were observed and a random draw from $U(0, 1)$ did not exceed a preset threshold value between $]0, 1]$. This threshold value implied the aggressiveness of the Red force, and several threshold values were explored. Usually, too passive Red force leads to the Blue force learning aggressive and bold tactics, while an active and aggressive Red force naturally encourages a polar opposite solution.

After the Blue force was trained to combat the predetermined Red force, the Red force was trained against it. If successful, two training rounds would lead to a good policy for both sides and could be tested. A similar solution has been shown to converge when training cooperating agents independently [93] in a decentralized setting. In this case, the training is centralized for both sides, but the sides are trained sequentially.

The test runs consisted of 1,000 episodes and produced the number of ships neutralized on both sides as well as victories achieved, respectively. All successful engagements of the tested side were stored in a heatmap as a firing position, as these would indicate which positions the agent preferred to station ships for engagements. The movement patterns can also be stored to visualize the paths taken to reach the positions for engagements, tactical picture compilation, or other actions.

The visualizations of the test rounds cluster the engagements so that they produce an easily understandable image of the found course of action. The plot shows the shortest route for each unit to travel to the nearest engagement position, which is a cluster centre of firing positions produced with k-means [33]. The visualization of the COA can then be analyzed with the numeric results, i.e. how many ships were lost, how many opponents were neutralized and how many victories each side achieved.

5.2 DDQN

Deep Q-networks (DQN) were introduced by Mnhi et al. [46] in 2013, producing impressive results in Atari games. In their proposed solution, a CNN is used as an action-value function, and it maps the observed state into action values. The Bellman equation is used as an iterative update for $Q_{i+1}(s,a) = \mathbb{E}[r + \gamma \max_{a'} Q_i(s',a')|s,a]$. The actions are selected according to the highest value. The action-value function is optimized by minimizing the loss function in Equation 11

$$L_i(\theta_i) = E_{s,a\sim p(\cdot)}\Big[y_i - Q(s,a;\theta_i))^2\Big], \tag{11}$$

which is essentially a Mean Squared Error (MSE) where the target $y_i = \mathbb{E}_{s,a\sim\mathcal{E}}\big[r + \gamma \max_{a'} Q(s',a';\theta_{i-1})|s,a\big]$ and $p(s,a)$ is a probability distribution over sequences of states and actions a and s. The θ_{i-1} network is held static as the *target network*, which evaluates the target values and is updated periodically according to the parameters of current policy θ during learning. The $\mathcal{E}$ is the *emulator* in Mnih et al. paper, meaning the Atari game emulator that produces the samples of the game. The gradient of the loss function as a differentiation w.r.t. the weights is

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a\sim p(\cdot);s'\sim\mathcal{E}}\Big[\big(r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) - Q(s,a;\theta_i)\big)\nabla_{\theta_i} Q(s,a;\theta_i)\Big].$$

As DQN was noted to overestimate the action values under certain conditions, van Hasselt et al. [94] proposed an update labelled as Double Q-learning (DDQN). The DDQN reduces the overestimations of DQN by separating the maximum operation in the target network into action selection and action evaluation. In essence, the target for the update is selected between the current policy and the target network as

$$Y_t^{\text{DDQN}} = R_{t+1} + \gamma Q(S_{t+1}, \text{argmax}\, Q(S_{t+1}, a; \theta_t), \theta_{t-1}).$$

Due to the results achieved by van Hasselt et al., DDQN was selected as the primary go-to version of DQN to test on the discrete version of the RL environment. The DDQN is described below in Algorithm 1. The used hyperparameters for DDQN are displayed in Table 1.

Hyperparameter	**Value**
Episodes	1,000-2,000
Time steps	40
Replay memory	10,000
Learning rate	0.0001
ϵ	$0.95 \rightarrow 0.01$
Decay	20,000
γ	0.95

Table 1: DDQN hyperparameters

Algorithm 1 DDQN implementation

Initialize θ and θ^{-}
Set learning rate λ, batch size b, total episodes t decaying ϵ for exploration and data buffer D = { }
while do episode $<$ episodes$_{m}ax$
    run episode, collect samples
    for $i \in n$ **do**
        actions = [] empty list
        for $agent \in agents$ **do**
            if $\epsilon < r_{\text{random}}$ **then**
                $a_t = \theta(o_t^{(a)})$
            else
                a_t = random actions
            end if
            actions + = a_t
        end for
        Execute actions for all agents, collect r_t, a, p and o_t
        data buffer += $[o_t, a_t, p_t, \hat{R}_t, v_t, g_t]$
        if episode % $k = 0$ **then**
            Create prioritized data loader w.r.t. $\hat{R}$
            Sample mini-batch $b_i \in D$
            Get y_i in b_i as $R_{t+1} + \gamma Q(S_{t+1}, \text{argmax}_a \; Q(S_{t+1}, a; \theta_t), \theta^{-})$
            Calculate $L_i(\theta_i) = E_{s,a\sim p(\cdot)}\left[y_i - Q(s, a; \theta_i))^2\right]$
            Adam update θ with backpropagation on L
        end if
        if episode % $m = 0$ **then**
            Transfer parameters $\theta \longrightarrow \theta^{-}$
        end if
    end for
end while

DDQN was tested in the described environment by first training the Blue side against the predetermined actions of the Red force. To test the effect of randomness, five random seeds were used in the initial setting of training, against the predetermined Red side. The training results are displayed in Figures 8 and 9.

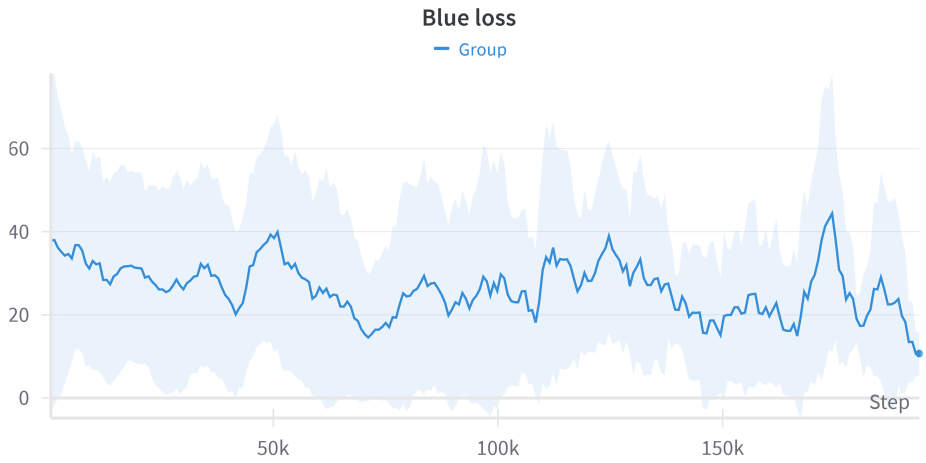


Figure 8: DDQN Blue loss in training with 5 random seeds

It was quickly discovered that it only required some 1,000 episodes for DDQN to find a policy, after which the results remained the same. For this reason, four of the training runs cover only half of that of the first two, which can be spotted Figures. The x-axis also displays a disproportionate number of steps due to parallel uploads to *Weights and Biases* [95], which was used to track the training data of all training runs.

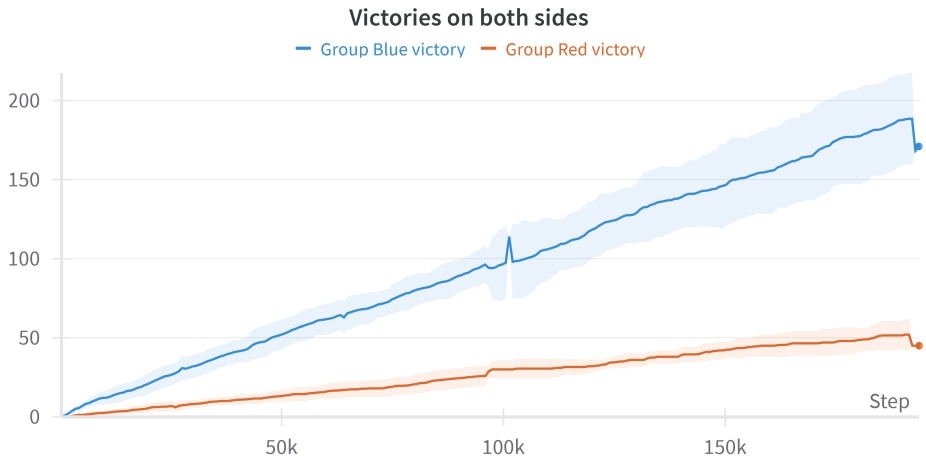


Figure 9: DDQN victory accumulation in training with 5 random seeds

Figure 8 displays the running average over 10 episodes and the standard deviation of the loss, while Figure 9 displays the unsmoothed mean and standard deviation of

the cumulative victories for each side during training. From these Figures, it is evident that the loss fluctuates, but the standard deviation decreases over the episodes, and the trend of the loss is downward, albeit slowly. Even though five random seeds are hardly a statistically relevant sample, it can be argued that for DDQN, the impact of randomness from the initial setting is minor. Instead, the dynamic environment and developed uncertainties cause difficulties in convergence.

In this setting, the Blue agent managed to come up with a policy that was victorious most of the time. Then, the Red agent was trained to learn a policy to counter the Blue agent's tactic. Due to a large state space and the focus of the reward function in the engagements, which are quite scarce, the Red actions were limited to be on average northbound for the first 20 steps of each episode so that the agents would spend less time learning the trivial navigation from South to the theatre and the training could focus on actions within the theatre. This could have been done by simply stationing the Red units in their respective operational area in the northern Baltic Sea, but this would have conflicted with allowing the Blue side time to reposition units before engagements.

However, the DDQN agent was eager to fixate on very simple policies. In testing, it seemed that the movement action space was too large for the agents to learn diverse movement patterns and instead, the network weights highlighted only a handful of movement choices for all occasions: Blue units might stay stationary for the whole episode while Red units might head directly South after reaching the operational area.

Figure 10 displays one COA found by the DDQN agent for both Blue and Red force. In Figure 10, it is apparent that the movement policy of a discrete agent is very rigid, as the heatmap of Blue engagements shows that all engagements are executed in very confined areas. The heatmap positions are visible under the yellow circles at the heads of each blue arrow. The result was identical for a 2 versus 2 scenario only with fewer units, so that figure is not displayed separately. Also, the heatmap is only displayed for one side at a time and has to be selected before starting the test runs. This is because the scale of the heatmap depends on the number of engagements, and if one side dominates the engagements, the other becomes invisible in the plotting.

Figure 10 displays the arrows for each participating unit from the starting point to the nearest engagement cluster centre. The centres are highlighted with a transparent yellow circle. In testing, the displayed tactic resulted in

	Blue	**Red**
Successful engagements	3114	1186
Victories	339	117

Table 2: DDQN COA numeric results

which means that Blue side was victorious 33.9 % of the time and the Red side 11.7 % of the time. The number of successful engagements exceeds the number of partaking Red units for the Blue side, which means there were engagements where several Blue units successfully fired the same target. This was supposedly prohibited in the

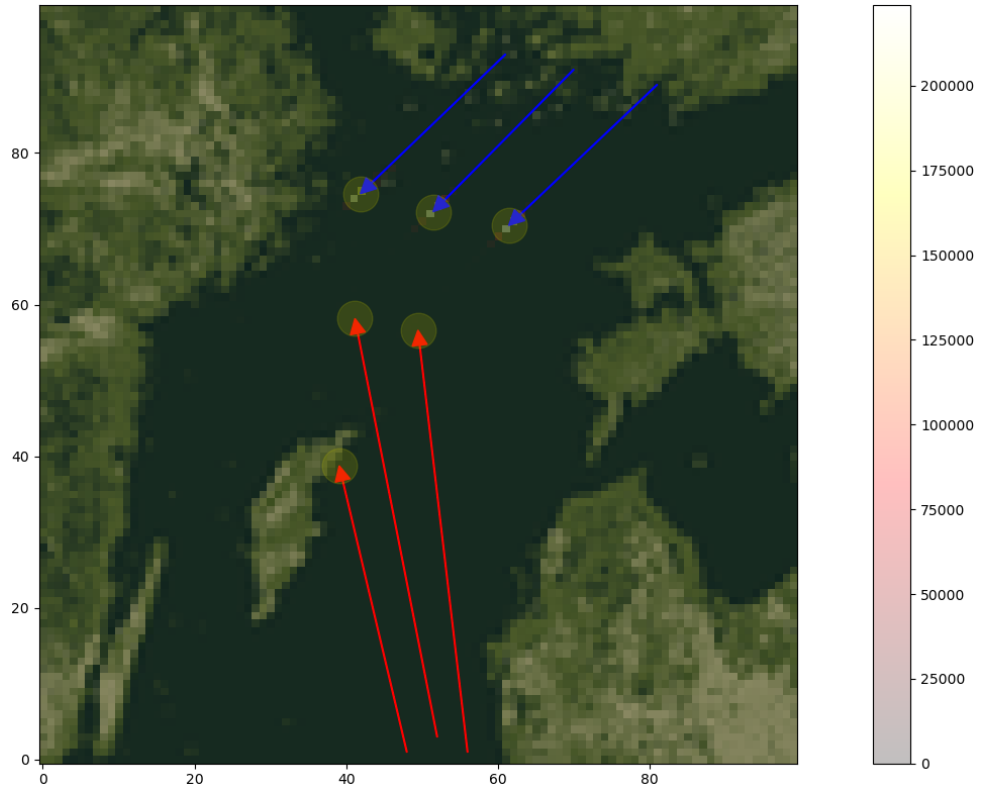


Figure 10: 3 versus 3 training results for one DDQN COA

environment, which means a bug in storing successful engagement data needs to be investigated further.

However, the DDQN agent demonstrates the ability to *solve* the environment to some extent. It can be argued that the found solution is quite crude and does not correlate with, e.g., Vego [54] principles of utilizing the Archipelago and harnessing the strengths of smaller combatants. Instead, it resembles a rather reckless head-on approach against a more formidable opponent. Intuitively and according to the principles mentioned in Section 3, this tactic seems futile. The larger vessels of the Red side are better off in the open sea, which intuitively should encourage a more subtle and cunning approach.

On the other hand, the smaller RCS and proper EMCON have, apparently, given the Blue side an upper hand in this approach. The impact of EMCON cannot be evaluated with the current design of the environment, and it should be added to enhance the tactical analysis. As such, the DDQN agent has at least provided an alternative COA to consider when planning the operation, although it can be argued that a discrete model of such a complex problem is unlikely to produce credible results.

5.3 Multi-Agent Proximal Policy Optimization

The PPO algorithm has been promising in multi-agent games [25], dubbed as MAPPO for Multi-Agent Proximal Policy Optimization. Due to the results showcased by Yu et

al. [25] it was selected as the primary algorithm to test with the continuous form of the environment.

In the implementation by Yu et al. [25], the MAPPO trains two separate neural networks, the actor and the value function or critic network. The networks can be shared between agents if the agents are homogeneous. In this environment, the observation state and action space can be set homogeneous for all surface combatants, so one actor network was suitable to be trained to function as the policy for all the agents on one side. Likewise, a shared value network was initialized. The value network can be agent-specific or agent-agnostic, and in this case agent-agnostic network was selected.

In Yu et al. [25] implementation the actor network π_θ is trained to maximize the objective function

$$L(\theta) = \Big[\frac{1}{bn}\sum_{i=1}^{b}\sum_{k=1}^{n}\min(r_{\theta,i}^{(k)}A_i^{(k)}, \mathrm{clip}(r_{\theta,i}^{(k)}, 1-\epsilon, 1+\epsilon)A_i^{(k)})\Big] + \sigma\frac{1}{bn}\sum_{i=1}^{b}\sum_{k=1}^{n}E\big[\pi_\theta(o_i^{(k)})\big],$$

where ratio

$$r_{\theta,i}^{(k)} = \frac{\pi_\theta(a_i^{(k)}|o_i^{(k)}}{\pi_{old}(a_i^{(k)}|o_i^{(k)})},$$

parameters θ are actor network parameters, b is the batch size, n is the number of agents, advantage $A_i^{(k)}$ is computed using GAE method [96], E is the policy entropy and $\sigma \in [0, 1]$ is the chosen entropy coefficient parameter, and $\epsilon \in [0, 1]$ is the scalar value used for clipping the ratio $r_{\theta,i}^{(k)}$. In the code implementation, as the objective function is to be maximized, it is implemented as a negation of $L(\theta)$ as the backpropagation and optimizer regard it as a loss function and update seeking to minimize its value.

The critic network V_ϕ, which maps the state to a reward as $S \rightarrow \mathbb{R}$, is trained to minimize the loss function

$$L(\phi) = \frac{1}{bn}\sum_{i=1}^{b}\sum_{i=1}^{n}\big(\max\big[(V_\phi(s_i^{(k)})-\hat{R}_i)^2, \mathrm{clip}(V_\phi(s_i^{(k)}), V_{\phi_{old}}(s_i^{(k)})-\epsilon, V_{\phi_{old}}(s_i^{(k)})+\epsilon)-\hat{R}_i)^2\big],$$

where ϕ are critic network parameters, $\hat{R}_i$ is the discounted reward-to-go and $V_{\phi_{\mathrm{old}}}(s_i^{(k)})$ are old values estimated by the critic network during collection of the data before the updates. The clipping is done to make sure the updates do not deviate too much from the earlier loss, i.e. to prevent exploding gradient problem.

In the implementation, MAPPO was implemented as shown in Algorithm 2 below.

The values and reward-to-go in algorithm 2 are normalized with Pop-Art [97] before updating θ and ϕ on $L(\theta)$ and $L(\phi)$ respectively. The standard deviation of the noise decays for the duration of training in a linear manner w.r.t. time steps executed to enforce adequate exploration in early-stage training but also to allow the actor to converge into a working policy.

As the engagements and especially victories are scarce events in the environment, prioritized experience replay [98] was used to give more emphasis on the samples that

Algorithm 2 MAPPO implementation

Initialize θ and ϕ
Set learning rate λ, batch size b, total time steps t and decaying standard deviation for noise σ
while do time steps < steps$_max$
  data buffer D = { }
  collect trajectories
  for $i \in b$ **do**
    τ_i= tensors$[batch, steps, agents, features]$
    for $s \in steps$ **do**
      actions = [] empty list
      for $ship \in ships$ **do**
        $a_t, p_t = \pi(o_t^{(a)})$
        actions = $\max(0, \min(actions + N(0, \sigma), 1))$
        Execute actions for all agents, collect r_t, a, p and o_t
        Concatenate and collect global states $g_t = \left[o_t^i \ \cdots \ o_t^n\right]$
        Estimate value $v_t = V_\phi(g_t)$
      end for
      Compute reward-to-go $\hat{R}$
    end for
    data buffer += $[o_t, a_t, p_t, \hat{R}_t, v_t, g_t]$
  end for
  Create prioritized data loader w.r.t. $\hat{R}$
  for $e \in$ epochs **do**
    Sample mini-batch $b_i \in D$
    Adam update θ on $L(\theta)$ with data b_i
    Adam update ϕ on $L(\phi)$ with data b_i
  end for
end while

lead to higher rewards instead of uniform sampling. The batch sampling was done with replacement enabled. The MAPPO hyperparameters are displayed in Table 3.

Hyperparameter	Value
Total time steps	100,000-200,000
Trajectories	10
Trajectory steps	40
Learning rate	0.0001
Batch size	64
Epochs	5
ϵ	0.2
Noise σ	[0.75, 0.05]
γ	0.99

Table 3: MAPPO hyperparameters

The initial implementation of the continuous action space and MAPPO proved to be inefficient in exploring the environment and finding reward signals unless the sensor ranges were made artificially longer. With long detection ranges, it was considerably easier for the agents to learn a working policy, but it did not reflect the reality as intended.

The exploration of the agents was investigated further both in theory and practice. Due to difficulties in exploiting discoveries enabled by larger action noise, it was deduced that it would be easier for the agent to learn favourable actions if the noise was instead fed into the network parameter space: it was deduced that this way, the noise is more *transparent* for the optimizer and gradient descent update, as a certain network structure leads to certain actions and the update has less guesswork when compared to adding random noise to the action outputs. It must be stated that this idea of parameter noise was discovered independently and perceived as novel in nature before previous research on the subject was searched for. However, previous research on the subject by Plappert et al. [38] establishes that "parameter noise outperforms traditional action space noise-based baselines, especially in tasks where the reward signal is extremely sparse". It has already been established that the reward signals of this warfare environment are sparse, so a version of parameter space noise was implemented.

There are several issues with parameter space noise as it may induce exploding gradient problem or lead to divergence instead of convergence. The amount of parameter space noise has to be adequate to aid exploration but manageable enough not to be counterproductive. In this case, parameter space noise was added as Gaussian noise $\sim N(0, 0.5)$ to the parameter space by clipping it into range $[-\kappa, \kappa]$, in which $\kappa = 0.05$ proved empirically to work well. For each episode, the actor network was reset to its noiseless version into which the clipped Gaussian noise was added into all layers except normalization. Layer normalization [99] ensured that the inputs were from the same distribution for the duration of the episode. The parameter space noise

was also made adaptive so that it was decreased if the agent managed to produce rewards and increased if no significant rewards were encountered. The ceiling of clipped parameter noise is the set $noise \sim N(0, 0.5)$ clipped to $[-0.05, 0.05]$, which was then reduced if agents performance was improved and slowly increased if the performance dropped.

The implemented parameter space noise policy aimed to take greedily advantage of discovered successful policies. If several victories emerged in the same rollout, the learning rate was decreased as $\frac{lr}{batch\ victories}$ while simultaneously cutting the total number of required time steps to avoid overfitting. Additionally, the training epochs for each rollout were increased by the number of victories achieved in the ten trajectories to amplify the successful policies' impact on training.

This approach proved successful in a 3 versus 3 scenario for the Blue side against the predetermined Red side as shown in Figures 11, 12 and 13. The resulting agents boasted a good performance against the predetermined Red side when using *aggressive* reward function, i.e. neglecting the negative reward of losing own units.

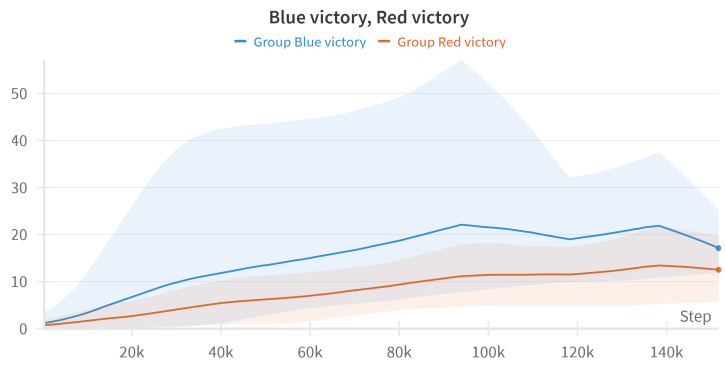


Figure 11: Victories of 6 random seed training rounds

Figure 11 displays the smoothed running average of victories over 40 steps, the mean and the interval between minimum and maximum values. It is of notice that one run stopped early at 100,000 time steps, as it consistently won the episodes, and therefore, the training came to an early stop. This shows as a decrease towards the 150,000 time steps for the Blue side. However, only one run was able to perform in such a manner.

Figure 12 displays the actor and critic loss of Blue agent during 6 training rounds. The outliers have been neglected and the Figure displays the median loss and its minimum and maximum intervals. As evident, the loss does not converge well and this implementation produces a rather unstable training process. This is due to the scarce rewards: if the agent does not win or manage to engage the opponent for multiple rollouts, the policy converges to smaller reward signals and then, when encountering scarce high rewards, the signals have a shock-impact on the value function as well as the policy. It can be argued that tuning the training scheme and increasing the training episodes could lead to better convergence, but due to the scarce reward mechanism a smooth convergence is unlikely. One option would be to increase the complexity

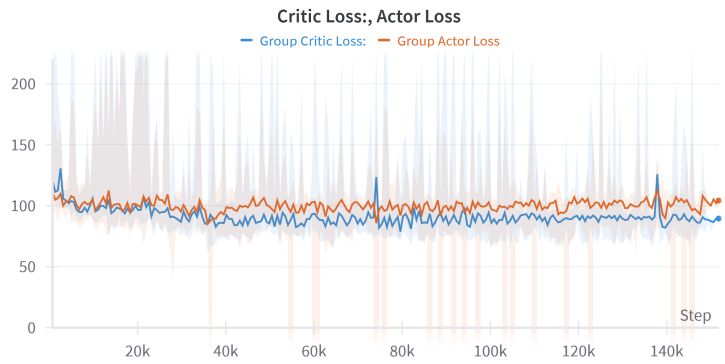


Figure 12: Actor and critic loss of 6 random seed training rounds

of the neural network and add memory such as an attention mechanism [100] to the training so that the scarce reward signals are not forgotten by the agent. However, the testing of different random seeds showed that the training results depend on the randomness of the developed environment, not only the random seed, as all learned policies exhibit similarities. The visualized tests show that all the agents have learned similar approaches in the given setting, which, together with the numeric analysis, indicates that the agent can converge into a solution by tuning the agents and reward. Diverse solutions are thus enabled by modifying the reward function or opponent policy.

During testing, a 1,000 episodes were run against the predetermined Red side for each trained Blue agent. The results are displayed in Figure 13.

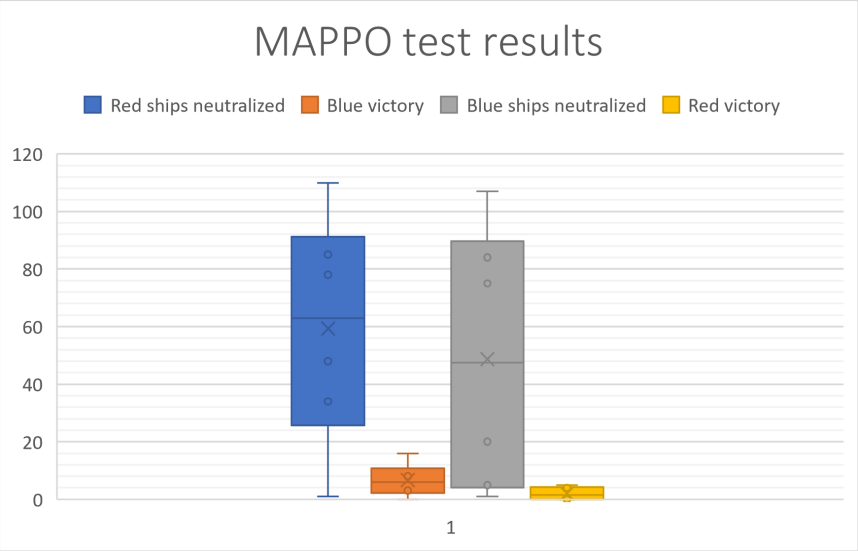


Figure 13: Test results of 6 random seed training rounds

Figure 13 displays the test results for 6 different random seeds. One seed produced a poorly performing agent, while the rest produced similar results between the agents. Visual inspection of the resulting tactics shows that, in this setting and reward function

formulation, the Blue side is prone to converge into a similar solution despite different random seed selections. On average, the Blue side manages to neutralize $\approx$ 60 Red units per 1000 test episodes and achieve seven victories against $\approx$ 49 losses and two Red victories. However, the standard deviation is quite high: 39.4 and 5.6 for successful Blue engagements and victories, respectively. If the lowest performing test run is neglected, the Blue results are 71 engagements ($\sigma^2 \approx 30.3$) and 8 victories ($\sigma^2 \approx 5.1$). After this, the Red side was optimized against the best-performing Blue side agent.

The resulting COA suggestion for Blue side is displayed in Figure 14.

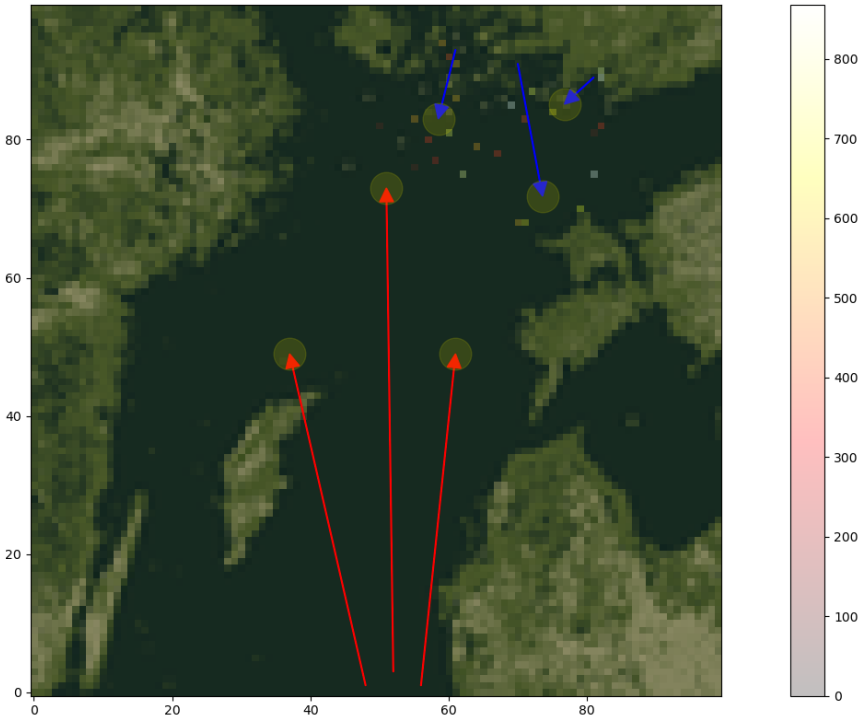


Figure 14: Blue side COA suggestion in a 3 vs 3 scenario

Figure 14 displays a COA suggestion as a visualization of the results of testing the trained Blue agent against the trained Red agent. The numeric results are displayed in the Table 4 below.

	Blue	**Red**
Successful engagements	25	5
Victories	3	1

Table 4: MAPPO COA numeric results

The numeric results are not quantitatively impressive for either side. However, with this COA, Blue side is five times as likely to win the encounter, inflicting almost eight times the damage compared to the Red solution, *in this particular setting*. The

actual reasoning behind both tactics is difficult to fathom, but the displayed Blue side profile complies with Vego [54] principles by utilizing the cover of Archipelago to some extent. Likewise, the Red COA highlights the task of assuming control of the area by spreading the force to the northern Baltic Sea. As such, the found solutions seem feasible and in-line with the original idea of the mission for both sides.

The Blue side tactic, as evident from the heatmap, is not very precise as the missile launch positions displayed by the heatmap are quite scattered; there are launch locations within and outside Archipelago. It is noticeable that the solution aims to keep one unit behind while sending two units to confront the enemy. Two units is the minimum number to perform fixing with EW bearings, so the tactic is sound in this sense. Likewise, the Red side can be interpreted to minimize risks by leaving two units behind and use one unit as the radar picket [101], a task that is supposed to increase the radar coverage to give an early warning of enemy units or, in this case, to expose enemy units for targeting. The same can be said about the Blue side tactic. Hence, both tactical schemes seem reasonable. The low number of combat actions needs to be considered, as the three cluster centers of the Red force are based on only 5 successful engagements: the results are not very valid in this sense.

The analysis of the results is a problem of its own. If the numeric results are used to formulate probabilities regarding the anticipated engagement, the frequentist probabilities are very low due to 1,000 test episodes. A better option could be to assess only the episodes in which an encounter took place, or re-run the test episodes with more time steps, for example. If only victories are considered, the Blue side has $\frac{3}{4} = 75$ % probability of winning. However, this percentage is quite irrelevant, as the encounters themselves are improbable as both sides have not converged to an aggressive solution that actively seeks engagement. A Bayesian analysis could best describe the results: given a probability of an encounter in testing as a prior probability, there is a posterior probability of winning or losing and, similarly the probability of losing units. In this particular case, there were mere 15 episodes with engagements, meaning that the prior probability of an engagement, given that both sides follow the policies displayed, is 1.5 %. This prior can be combined with the posterior probability of inflicting damage or sustaining it to produce a more thorough analysis. Another aspect to consider is the Bayesian utility. If a solution is likely to result in a victory but also likely to result in losses, the utility should be defined by the decision maker. In this case, the utility would be the cost-reward ratio between the victory or achieving the mission goal and losing units.

Another produced COA, with an identical starting point and against the predetermined Red force, is displayed in Figure 15. In this COA, the Blue agent spreads the units out to the open sea to a greater extent. Different versions and iterations of the same setting were tested and tuned, also training Blue agents from scratch to counter the trained Red force. Two main categories of solutions stand out: meeting the opponent at the open sea or holding at least some units either within or in the vicinity of the Archipelago. The latter tactic correlates with, e.g., Vego's principles of littoral warfare [54].

The numeric results for this COA, displayed in the table below, reflect the fact that positioning units within the Archipelago place the Red side at a disadvantage but

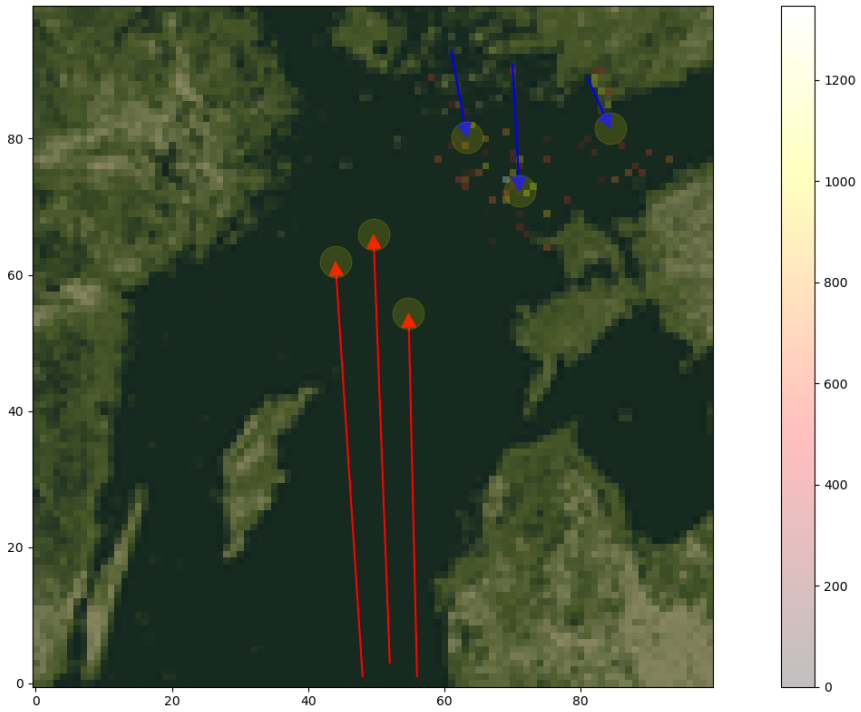


Figure 15: Blue side COA 2 suggestion in a 3 vs 3 scenario

simultaneously greatly reduces the likelihood of the encounter altogether, at least in this setting.

It is noticeable that a defensive COA, although having a small likelihood of taking place when the numbers are compared to total test episodes, is the most favourable in preserving Blue units. In this instance, the decision could be to keep units close or within the Archipelago and use some other asset for targeting instead.

5.4 Comparison

According to the previous subsection, it can be argued that MARL can be utilized to produce COAs to support decision-making. Albeit the discrete version produced, in one instance, more profound numeric results, as displayed in Table 2, the found solution seems too optimistic. Additionally, the proposed COA of Figure 10 is very simplistic for both sides. According to the visualization, it can be argued that the policy converged to an adequate local minimum, producing adequate results as the same simplicity held back the opponent. The third Red unit did not manage to enter the actual operation area, which most likely affected the results to a great extent, though staying on the Eastern side of Gotland most likely protected it to some extent.

The MAPPO agents produced considerably better results. As the implementation maps the observation space into normal distribution mean and standard deviation, from which the actions are drawn as random samples, the procedure promotes a more diverse set of results to perform inference and decision-making. While the numeric results fall short of those exhibited by the DDQN agent, it can be easily

argued that the rather linear approach of DDQN policies would not work as well in the real-world setting. As such, the sole numerical results cannot be compared between the algorithms.

In conclusion, two different algorithms were tested, and the MAPPO performed better in the sense that it produced more variety in tactics and displayed more soluble policies. The continuous action space deals better with uncertainties and enables a more diverse approach to the problem solution. Likewise, parameter noise made the training of MAPPO effective and sample efficient. Utilizing trajectory batches for training also allowed to exploit success better than the replay memory of DDQN. However, the uncertainties related to the MAPPO agents caused the environment to be very dynamic when both sides followed their policies. With a predetermined policy on one side, the agents were comparably quick to learn a solution, as the opponents' movements followed a predictable pattern. When this changed, the results were mediocre at best. This finding highlights the need for a more sophisticated approach and rethinking, e.g., the reactive policy and deep RL architecture.

6 Conclusion

This chapter provides a summary of results and findings and discusses them further to provide a critical look into the work that has been done and how it can be extended to produce further value.

6.1 Summary of results and findings

This thesis has produced several results. These can be categorized as:

- Determination of the objectives of naval operations in general
- An analysis of the impacting features in naval littoral warfare and ways to implement those
- A MARL environment for Littoral Surface Warfare
- A demonstration of solving a tactical scenario with MARL
- Producing alternative courses of action for the decision-maker with MARL

It can be immediately stated that the algorithm and agent testing have not been all-encompassing. Several other algorithms could have been tested, but these would have extended the work of this thesis beyond its original scope.

The purpose of this thesis is to follow OR methodology to formulate a problem and solve it to be able to enhance the related decision-making. The first step was to identify the major factors in littoral surface warfare. With the focus on the surface combatant, this ruled out several other platforms that have been deemed crucial [57, 54] such as conventional submarines and aircraft. These features can be implemented into the environment later to provide insight into joint littoral operations. With the focus on surface combatants, physical realities such as navigation and tactical picture compilation were studied to create a formal, computational representation of the environment. Once the affecting factors had been identified they were implemented into a program that can be found at https://github.com/valauri/Littoral-Naval-Warfare-MARL.

To demonstrate MARL in the environment, DDQN was quick to demonstrate numerically good but tactically poor results. This was expected, as the environment is complex and contains a cascade of probabilistic variables that make it non-stationary, on top of the non-stationarity produced by multiple agents. MAPPO, on the other hand, produced quite good results.

Regarding the results produced by MAPPO, the working policy for the Blue side reflected certain tactical principles of the Finnish navy and Vego [54]: some Blue units remained within or in the vicinity of the Archipelago to target the open sea and fired missiles from within. This tactic proved as a low-risk alternative as reported numeric results show, and could have provided the decision-maker with additional information to confirm one's choice of tactics.

The working MAPPO policy aimed to hide one or two Blue units near the Archipelago and target the opponent with one or two units that acted as radar pickets

of sorts [101]. However, the primary tactic was to use two units to target the Red units and fire missiles on contact, followed by a salvo from the Blue unit that stayed behind. On average, the tactic was adequately successful in engaging the Red units. As such, both tactics found by Blue agents can be seen to fit the common understanding of naval tactics.

Due to non-stationarity and resembling challenges in RL overall, the results of training are quite unpredictable and, in this environment, depend heavily on setting the agent to an optimal start. If the agent begins the exploration from the wrong setting, the policy is unlikely to converge to a good solution in the training process. This issue relates to reward engineering, but despite, e.g. encouraging agents to extend their search areas, the result may be that the agent is unable to react to the observations (enemy sightings) correctly, whereas the opposing side may be ready to inflict losses and therefore the agent concludes that the found state produces losses instead of gains in reward. This was addressed by utilizing different reward functionalities for different phases of the operation, namely *aggressive* and *defensive* profiles. The difference between the two is that *aggressive* reward calculation does not take losses into consideration and thus favours inflicting damage over received damage. However, the impact of these two options was not tested thoroughly.

The non-stationarity was tackled with several solutions, of which one was the design choice to keep the starting positions for 2 and 3 units always identical. This should reflect the starting situation of the planning process, so for a different scenario, these would need to be modified. Another solution that would reduce non-stationarity and guide the exploration is to reduce the environment during early phase of training and expand it during training. In this particular problem, the starting positions could have been closer to the center of the map in the early stages of training and extended outwards to the desired starting points later on, or the whole game grid could be reduced to a smaller area in the early stage.

Overall, the results indicate that RL and MARL are able to produce COAs and aid in evaluating them, but the use of RL and MARL shifts the planning in headquarters to reward engineering in order to produce courses of action that correlate with the commander's intent of performing the operation. Likewise, to produce an effective number of COAs, the agent has to be trained extensively with different reward signals and settings to create variety in the results and to explore the applicable tactics further. If this is not feasible due to time constraints, a quick standalone solution can also be found if the other features and variables are taken into consideration properly.

6.1.1 Answers to research question

The theoretical outlook in sections 2 and 2.4 on AI and RL briefly introduced RL to establish its core ideas and methodologies. The following Section 3 established an understanding of the basic principles of naval warfare and littoral warfare, from which the minimum essential aspects were driven to implement the RL environment. The design choices are open for discussion and, as stated above, do not reflect real-life specifications due to sensitive data and classified information but serve only as an example to demonstrate the use of RL in this context.

Originally, the idea was to test at least three RL algorithms to solve the environment. As the environment proved to be a difficult learning problem for the agents, even two algorithm implementations took a considerable amount of time in testing different implementations in the environment and agent configuration. Due to this, the testing was limited to two algorithms. Between the two, DDQN and MAPPO, the continuous action space enabled by MAPPO quickly established itself as the primary solution able to solve the environment to an acceptable extent. It can be argued that imitation learning would have helped the agents to learn the proper state-action pairs faster instead of mere trial and error, but imitation learning may introduce further operator bias to the result and thus interfere with the results if the goal is to test whether an AI is able to find novel solutions that supplement the existing doctrine. However, this work has laid out the fundamentals in drafting a model of the littoral warfare environment and, to some extent, managed to solve it, although a lot of finishing work was left for future research.

For the final research subquestion, it can be stated that all the working policies concurred with existing prerequisites and principles of littoral surface warfare. The reasons for this are discussed in the next section. On a technical note, it must be stated that the stochastic nature of exploration is a major issue in utilizing RL in tactical decision-making in said manner, as the learning results depend on chance and certain combinations of sequential actions happening and succeeding to reinforce a certain policy. Thorough training and testing are required to make sure that the resulting models have comprehensively investigated the possibilities in the state-action space to provide alternative solutions and evaluate these depending on, e.g., the risk profile of each tactic. For example, the radar picket tactic displayed itself as a viable solution but required the decision maker to accept the high risk of losing one unit i.e. half of the available assets. The utility of such as decision is the value of a victorious encounter, but it may hinder the ability to execute the next mission and is likely to result in a loss of lives that could be avoided with a more defensive approach. However, from the research point of view, the agents were able to produce alternative tactics for the decision-maker to choose from, which answers the research question w.r.t. the constraints of the implementation.

6.2 Discussion

Regarding the research overall, the sources used are mainly of high quality and relevance. The source material on AI and naval warfare is well-established, composed of scientific papers and books. The two theses that have been used as source material can be argued to be the least relevant and credible, but as explained, the information used from these theses only serves to provide certain baseline values to implement the littoral scenario for demonstrating MARL, and as such, the actual and cross-examined probabilities and ship capabilities are not a key element. For the ship references, mere online sources were cited as those are easily accessible, unlike more thorough books such as *IHS Jane's Fighting Ships*.

As explained in the previous section, the results of this research indicate that littoral naval warfare can be formulated as a partially observable stochastic game in

which multi-agent reinforcement learning can be used to verify selected tactics by approximating a working tactical scheme, i.e. policy, for a particular environmental setting. While some tactics were discovered by MARL, even the unorthodox ones were not completely new in the context of littoral or naval warfare. The purpose of this research was to test if MARL can be utilized to support the decision-maker at the squadron level, and as such it seems that using a rather simple environment and easily accessible computational resources a quick tactical analysis can be performed to provide further insight into the planning and execution of littoral naval operations.

Discovering completely new tactical solutions in the built environment would have been interesting. The reasons for the lack of innovative solutions may be due to the author's own bias, which is transferred into the implementation of the environment: the environment reflects the subjective perception of the problem and may, therefore, favour solutions known to the author. To combat this issue, the implementation should be cross-examined using a suitable method, for example, an expert panel that would give consolidated input on the features of the implementation. Likewise, if developed into a playable game, the experts could play the encounter to produce a baseline of performance from a human perspective for the MARL to imitate and improve on.

As an example of completely novel solutions, it would have been interesting to see the Blue agents, for example, aim to flank the Red force by sending units West-South West and thus perform in an unanticipated manner. Such behaviour could result if the training times were longer and the Blue force found the cover of the Archipelago unfavourable. It can be speculated that aerial reconnaissance could expose the units within the Archipelago and encourage Blue force to evade the incoming Red engagement in the manner described.

The research report already discusses the shortcomings of the features of the implementation, as there is technically an unlimited number of features to implement and variations in their implementations. However, a selected few can be discussed here. The first is the lack of other-than-surface units. For example, no aerial assets are implemented, nor subsurface assets. To be useful as a decision-making support system, the multitude of littoral assets should be implemented with respective capabilities to reflect the real-world environment of littoral warfare more accurately. These will be implemented in the future, as this research is supposed to continue as doctoral studies in operations research regarding the possibilities of unmanned systems in littoral warfare.

The second aspect is the grid-based solution with discrete time steps. The grid was originally selected to make the environment easier to handle from the developers' and agents' perspectives. To a minor extent, it also resembles how naval assets are coordinated: surface vessels are usually given respective areas in coordinates or in a pre-determined grid. The exact position of the ships does not need to be micro-managed, and therefore, more vague coordination is in order. On the other hand, the less specific environment hinders the ability to make use of the terrain. Now, the crude map does not reflect very accurately the altitude and depth differences characteristic of littoral warfare. However, the agents were able to make use of the terrain nonetheless.

The way the environment was implemented with discrete time steps also imposes

difficulties in the execution of the actions. Now the actions were selected for each unit before execution to make the decision-making situation (state) identical. Then the units performed the actions so that movement and EMCON alterations were executed immediately while engagements were stored so that two units could have a successful engagement on each other in the same time step, i.e. neutralize one another simultaneously. This would have worked better in a continuous time environment.

For the deep learning implementation, a normal distribution was used as the sample distribution for action selection. Other distributions were tested as well: categorical and beta distribution. Intuitively, beta distribution should have worked better than the normal distribution, as there is no need to clamp the result into the $[0, 1]$ range in which the continuous version of the agent operates. Despite this intuition, the beta distribution version performed poorly. To test the categorical distribution, it was implemented for engagement actions and EMCON control: the categorical distribution would produce the probability to pick salvo size or the binary on-off selection for radar, while movement actions course and speed were still determined with a beta distribution. This solution performed equally poorly in early testing and was therefore abandoned, and the effort was put into the normal distribution-based solution. The alternative distributions could be experimented on again with a more refined environment, but as the normal distribution was able to produce working policies, this was deemed unnecessary.

The neural network structure also underwent a lot of experimenting. The early solution consisted of strictly linear layers: the input was size $7 \times 7 + 4 \times n + 3$ in a single dimension, as it covered the nearby area of the agent, the states of all cooperating units n for which their state was $s_i = [x, y, m, r]$, including the position (x, y), number of missiles left m and EMCON state $r \in [0, 1]$, as well as number of found targets, *proximity to closest replenishment point* and the ducting factor. The ducting factor was added to give the agent some perception of the range of its sensors, i.e. the same position might not be worth the same value with a different ducting factor. It was deemed, however, that the nearby environment covered too much of the input vector and the hypothesis was that it might make it more difficult for the agent to learn the meaningful features, such as when to engage when there were so many grid values in the input. Thus, the network was modified to separate the grid values, reshape them into a 7×7 two-dimensional grid and feed them through a CNN to produce a more compact description of the surroundings. The resulting 8 values were fed to a linear layer that projected them into 12 values, reducing the input vector to 25 values in total. Later, the distance to replenishment point was changed to ship type (0 for combatant, 1 for a landing ship), because the episode length did not support actually replenishing depleted missiles. The 25 values were then fed to a linear feedforward network to produce the mean and standard deviation values for each of the four actions, which worked best in this case. The number of layers and neurons was also experimented on, with the common observation that a large network tends to be more unstable to train and cannot generalize into the deviations in the non-stationary environment compared to a simpler model. It can be argued that a large enough model could learn the state-action pairs more accurately, but as valuable samples are scarce, it would require extensive exploration and consume considerably more computational

resources.

When comparing the chosen solution and performance to the sophisticated and exquisitely performing AlphaStar [9] the lack of memory is evident. The reactive policy of agents has been brought up in the thesis and while simple to implement, the lack of memory seems to hinder the performance of the agents to a great extent. Therefore it can be argued that the performance could be increased dramatically by introducing a memory to the decision-making policy, in the form of LSTM or a transformer, to guide the agents decision-making process as has been done with AlphaStar. In short, it can be said that the reactive policy is insufficient in producing robust results despite being able to conjure adequate policies.

If a proper visual representation of the environment had been created with a game engine, the obvious solution would have been to use a deep CNN to grasp the information directly from the screen, as was done with e.g. AlphaStar [9]. The lack of a game engine-empowered interface and visual representation can be seen as a major drawback of the implementation, as observing the agent's behaviour becomes tedious. The implemented way to visualize the combat is, first of all, computationally cumbersome and visually not very convincing. To apply these methods in practice, a solid visual representation is in order to make the evaluation of the results more transparent.

Despite these mentioned issues and shortcomings, of which only the foremost are discussed, the thesis has demonstrated the use of multi-agent reinforcement learning in supporting littoral naval warfare operation decision-making on a tactical level, using only one person's time, effort and knowledge and one laptop with 3070i RTX GPU. With these resources, the training time for one side and 250,000 time steps was approximately 80 minutes, so dual-sided training took 160 minutes, and more comprehensive testing with several random seeds would take less than 12 hours. It is noticeable that the implementation does not use parallel computing and only $\sim$ 30 % of GPU capacity was used at a time: implementing parallel computing could drop the training times on a single working station considerably; being able to distribute the computation in 3 parallel processes could enable training both sides once in an hour on the used device. Now the parallel computing was implemented by manually starting parallel runs. It is also noticeable that the environment is not optimized for performance: the code is crude and lacks finesse. Despite these facts, the implemented simulation and relatively non-complex agents were able to produce results with minimal resources. Even complex improvements in implementation would not make the on-premise use of such models infeasible for squadron command but rather allow for a 24-hour planning schedule while running a common war game scenario for alternative options in due time before actual execution. In this sense, the research can be deemed successful in demonstrating the use of AI in decision-making support in the context of littoral naval warfare.

As previously mentioned, the purpose is to develop this environment further to serve as an OR tool to assess and examine, e.g. the impact and possibilities of the gradually increasing number of unmanned and autonomous naval assets in the littoral context.

6.3 Research Contribution

As discussed in previous subsections, the results of this thesis demonstrate the use of MARL as a decision-making support tool for tactical decision-making by creating and aiding in the evaluation of different COAs. The demonstration is not meant to be exhaustive, i.e. produce actually usable tactics or a production-ready program to be utilized. Instead, the thesis has established that with an analytical approach, using OR methodology to formulate a problem into a computable form, tactical decision-making can be enhanced with MARL by making the planning process more effective and pervasive time and solution-wise.

The work done to achieve the results of the thesis outlines several features that need to be taken into consideration when developing a MARL solution to support tactical decision-making. To function as an actual support system, either an existing command and control system would have to support the use of MARL as a simulation platform, or a more sophisticated environment with the correct coordinate system, topography, and unit capabilities needs to be developed. However, the displayed principles remain the same. The findings of this research indicate that such a system would have to

- Realistically formulate the combat features
- Include optional algorithms for testing
- Include options for reward engineering to
  - Encourage desired tactical approach i.e. *defensive*, *offensive*
  - Enforce known limitations such as maritime borders
  - Utilize pre-trained models to speed the training process
- Utilize network parameter noise due to scarcity of rewards in combat
- Have a proper GUI for human-machine interaction to enable evaluation and imitation learning
- Utilize parallel computing to speed up the training process
- Implement a memory-based or hybrid policy instead of solely reactive policy
- Create a tool to analyze the results w.r.t. chosen utility

The combat features displayed here do not resemble reality *per se*, as explained in Section 4, so the user would have to be able to insert the actual values and probabilities provided by, e.g., the intelligence sector or to interpolate a consensus from available experts.

The developed environment, as in many RL applications, is susceptible to random seed initialization, which may hamper or enhance the training results in an unexpected manner. The training of agents has to be run several times to ensure result relevance and consistency. However, in this particular use case, if a sound tactical solution is found *by chance*, its value is not reduced by the fact that it is an inconsistent outlier.

The reward engineering should be developed to encompass several ready-made and tested options so that the user can choose a desired profile instead of rewriting the program code to suit individual, observed needs. This way, the user experience would be enhanced, and the usability of the tool would be greater and low-threshold.

Pre-trained models that exhibit certain qualities would be useful to speed the training process in the desired direction. However, this may hamper the original idea of opposing user bias. Another option would be to use an ensemble method of several pre-trained agents to select optimal solutions, or to aggregate a selection of models into one global model that is presumed to exhibit the desired behaviour. Otherwise, certain limitations, such as maritime borders, would need to be considered to ensure the proposed COAs are feasible.

Network parameter space noise was discovered to be far superior to action noise in training the agents. The issue with network parameter space noise is that it needs to be added in a correct proportion so as not to induce exploding gradient problems. The use of parameter noise needs to be managed so that its impact is decreased or increased proportionally to the results achieved by each episode.

The lack of a proper GUI, HMI (human-machine interface) and visualization tools was denoted as problematic earlier, and real-life applications would need to be addressed to make the use of MARL more explanatory. This is a crucial aspect, as the decisions supported are combat-related, so the decision-maker has to be able to justify the inference chain that has led to a particular decision. More so than in many other fields that require decision-making.

Additionally, the simple reactive policy showcases itself as insufficient in producing robust results, and therefore it is suggested to use a memory-based or a hybrid solution to employ MARL solutions into warfare settings. Likewise, the proper analysis of different outcomes requires a tool such as suggested Bayesian analysis of the utility of a certain course of action. The Bayesian utility can be used to assess the estimated outcome to its cost and therefore aid the decision-maker in evaluating the expected impact of the decision.

In summary, this thesis has contributed by demonstrating the use of MARL in naval tactical decision-making support and by identifying aspects that need to be taken into consideration when building a reinforcement learning-based tactical decision-support system. The thesis has also paved way to utilize MARL in evaluating performance of different naval assets, as the used methodology and similar environment can be used to evaluate e.g. the impact and possibilities of deploying uncrewed maritime systems.

References

[1] S. Tangredi, G. Galdorisi, AI at War: How Big Data, Artificial Intelligence, and Machine Learning Are Changing Naval Warfare, Naval Institute Press, 2021.
URL https://books.google.fi/books?id=ioITEAAAQBAJ

[2] L. Vasankari, Tekoäly ja automaatio tulevaisuuden laivastojoukoissa (2022).
URL https://urn.fi/URN:NBN:fi-fe2022080953410

[3] Falkland islands war (2023).
URL https://www.britannica.com/event/Falkland-Islands-War

[4] R. Pedrozo, The russia-ukraine conflict: Blocking access to the black sea, Naval War College Review 75 (4) (2022) pp. 37–52.
URL https://www.jstor.org/stable/48731997

[5] Y. Shoham, K. Leyton-Brown, Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations, Cambridge University Press, Cambridge, UK, 2009.

[6] R. S. Sutton, A. G. Barto, Reinforcement learning: An introduction second edition, MIT press, 2018.

[7] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, D. Hassabis, Mastering the game of Go with deep neural networks and tree search, Nature 529 (2016) 484–489. doi:10.1038/nature16961.

[8] C. Metz, In two moves, AlphaGo and Lee Sedol redefined the future (March 2016).
URL https://www.wired.com/2016/03/two-moves-alphago-lee-sedol-redefined-future/

[9] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. P. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, D. Silver, Grandmaster level in StarCraft II using multi-agent reinforcement learning, Nature (2019) 1–5.

[10] Command: Modern Operations (2019).
URL https://www.matrixgames.com/game/command-modern-operations

[11] W. P. Hughes Jr., A salvo model of warships in missile combat used to evaluate their staying power, Naval Research Logistics (NRL) 42 (2) (1995) 267–289. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/1520-6750\%28199503\%2942\%3A2\%3C267\%3A\%3AAID-NAV3220420209\%3E3.0.CO\%3B2-Y, doi:https://doi.org/10.1002/1520-6750(199503)42:2<267::AID-NAV3220420209>3.0.CO;2-Y.
URL https://onlinelibrary.wiley.com/doi/abs/10.1002/1520-6750%28199503%2942%3A2%3C267%3A%3AAID-NAV3220420209%3E3.0.CO%3B2-Y

[12] M. J. Armstrong, A stochastic salvo model for naval surface combat, Operations Research 53 (5) (2005) 830–841.
URL http://www.jstor.org/stable/25146917

[13] M. J. Armstrong, Effective attacks in the salvo combat model: Salvo sizes and quantities of targets, Naval Research Logistics (NRL) 54 (1) (2007) 66–77. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/nav.20187, doi:https://doi.org/10.1002/nav.20187.
URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.20187

[14] T. Dupuy, The Quantified Judgment Method of Analysis of Historical Combat Data: A Monograph, Historical Evaluation and Research Organization, 1974.
URL https://books.google.fi/books?id=vzzkGwAACAAJ

[15] E. M. Mooren, Reinforcement learning applications to combat identification (2017).
URL https://apps.dtic.mil/sti/citations/AD1045940

[16] J. Wang, J. Wang, J. He, G. Wang, M. Wang, Research on naval air defense intelligent operations on deep reinforcement learning, in: 2022 34th Chinese Control and Decision Conference (CCDC), 2022, pp. 2246–2252. doi:10.1109/CCDC55256.2022.10034115.

[17] M. Rempel, J. Cai, A review of approximate dynamic programming applications within military operations research, Operations Research Perspectives 8 (2021) 100204. doi:https://doi.org/10.1016/j.orp.2021.100204.
URL https://www.sciencedirect.com/science/article/pii/S2214716021000221

[18] A. Sztykgold, G. Coppin, O. Hudry, Dynamic optimization of the strength ratio during a terrestrial conflict, in: 2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning, 2007, pp. 241–246. doi:10.1109/ADPRL.2007.368194.

[19] V. N. William Kemple, Gary Porter, Littoral warfare simulation experiment, Proceedings of the 1996 Command and Control Research and Technology Symposium: Command and Control in the Information Age (1996) 536–543.

[20] Occam's razor (2023).
URL https://www.britannica.com/topic/Occams-razor

[21] P. M. Morse, G. E. Kimball, Methods of Operation Research, The MIT Press, 1959.

[22] J. Bagby, R. Bowler III, T. Burnett, G. Marlowe, W. Mylander, J. Sigler, E. Smyth, Naval Operations Analysis, NAVAL INSTITUTE PRESS, 1989.

[23] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, R. Wirth, Crisp-dm 1.0 step-by-step data mining guide, Tech. rep., The CRISP-DM consortium (August 2000).
URL https://maestria-datamining-2010.googlecode.com/svn-history/r282/trunk/dmct-teorica/tp1/CRISPWP-0800.pdf

[24] H. van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double q-learning (2015). doi:10.48550/ARXIV.1509.06461.
URL https://arxiv.org/abs/1509.06461

[25] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, Y. Wu, The surprising effectiveness of ppo in cooperative, multi-agent games (2022). arXiv:2103.01955.

[26] J. Boyd, G. Hammond, A. U. U. Press, A Discourse on Winning and Losing, Air University Press, Curtis E. LeMay Center for Doctrine Development and Education, 2018.
URL https://books.google.fi/books?id=B-aVuQEACAAJ

[27] S. J. Russell, P. Norvig, Artificial Intelligence: a modern approach, 3rd Edition, Pearson, 2010.

[28] R. Bellman, An introduction to artificial intelligence : can computers think? / Richard Bellman, Boyd & Fraser Pub. Co San Francisco, 1978.

[29] E. Rich, Artificial Intelligence, McGraw-Hill, Inc., USA, 1983.

[30] A. M. Turing, On computable numbers, with an application to the Entscheidungsproblem, Proceedings of the London Mathematical Society 2 (42) (1936) 230–265.

[31] I. J. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, Cambridge, MA, USA, 2016, http://www.deeplearningbook.org.

[32] A. Jung, Machine learning: The basics (2022). arXiv:1805.05052.

[33] C. C. Aggarwal, Data Mining: The Textbook, Springer, Cham, 2015. doi:10.1007/978-3-319-14142-8.

[34] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (5) (1989) 359–366. doi:https://doi.org/10.1016/0893-6080(89)90020-8.
URL https://www.sciencedirect.com/science/article/pii/0893608089900208

[35] T. Sanger, P. N. Baljekar, The perceptron: a probabilistic model for information storage and organization in the brain., Psychological review 65 6 (1958) 386–408.
URL https://api.semanticscholar.org/CorpusID:12781225

[36] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors, nature 323 (6088) (1986) 533–536.

[37] R. S. Sutton, A. G. Barto, Reinforcement Learning: An Introduction, 2nd Edition, The MIT Press, 2018.
URL http://incompleteideas.net/book/the-book-2nd.html

[38] M. Plappert, R. Houthooft, P. Dhariwal, S. Sidor, R. Y. Chen, X. Chen, T. Asfour, P. Abbeel, M. Andrychowicz, Parameter space noise for exploration (2018). arXiv:1706.01905.

[39] R. Bellman, Dynamic Programming, 1st Edition, Princeton University Press, Princeton, NJ, USA, 1957.

[40] R. J. T. Bradley Efron, An introduction to the bootstrap, Chapman & Hall/CRC, 1993.

[41] R. Sutton, Learning to predict by the method of temporal differences, Machine Learning 3 (1988) 9–44. doi:10.1007/BF00115009.

[42] C. J. C. H. Watkins, Learning from delayed rewards, Ph.D. thesis, King's College, Oxford (1989).

[43] C. J. C. H. Watkins, P. Dayan, Q-learning, Machine Learning 8 (3) (1992) 279–292. doi:10.1007/BF00992698.
URL https://doi.org/10.1007/BF00992698

[44] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, P. Abbeel, Trust region policy optimization (2017). arXiv:1502.05477.

[45] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms (2017). arXiv:1707.06347.

[46] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing atari with deep reinforcement learning (2013). arXiv:1312.5602.

[47] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning (2019). arXiv:1509.02971.

[48] S. V. Albrecht, F. Christianos, L. Schäfer, Multi-Agent Reinforcement Learning: Foundations and Modern Approaches, MIT Press, 2024.
URL https://www.marl-book.com

[49] K. D. Sven Gronauer, Multi-agent deep reinforcement learning: a survey, Artificial Intelligence Review 55 (2022). doi:10.1007/s10462-021-09996-w.
URL https://doi.org/10.1007/s10462-021-09996-w

[50] A. T. Mahan, The influence of sea power upon history, 1660-1783 / Alfred Thayer Mahan, Methuen London [England], 1965.

[51] J. S. Corbett, Some principles of maritime strategy, Longmans, Green London, 1911.

[52] Finnish navy (2023).
URL https://merivoimat.fi/en/about-us

[53] V. Vänskä, Nykyaikainen merisodankäynti, Julkaisusarja, Maanpuolustuskorkeakoulu, Sotataidon laitos, Helsinki, 2021.
URL https://urn.fi/URN:ISBN:978-951-25-3200-1

[54] M. Vego, On littoral warfare, Naval War College Review 64 (2) (2015).
URL https://digital-commons.usnwc.edu/nwc-review/vol68/iss2/4

[55] D. F. Tver, Ocean and marine dictionary / David F. Tver, Cornell Maritime Press Centreville, Md, 1979.

[56] Sensing in clutter: Improving littoral situational environment (April 2009).
URL www.janes.com

[57] M. Vego, Naval Strategy and Operations in Narrow Seas, Taylor & Francis Group, 2003.

[58] HEADQUARTERS, DEPARTMENT OF THE ARMY, The Operations Process, Headquarters, Department of the Army, 2019.
URL https://armypubs.army.mil/

[59] B. Johnson, J. Green, G. Burns, T. Collier, R. Cornish, K. Curley, A. Freeman, J. Spears, Mapping artificial intelligence to the naval tactical kill chain, Naval Engineers Journal 135 (2023) 155–166.

[60] B. J. Christiansen, Littoral combat vessels analysis and comparison of designs (2008).

[61] T. A. Lopmeri Pekka, Suomen laivasto 1968-2003 osa 3, Meriupseeriyhdistys ry, Suomi Merellä-säätiö, 2008.

[62] Finnish Navy, Rauma class (2023).
URL https://puolustusvoimat.fi/en/equipment#/asset/view/id/332

[63] Finnish Navy, Hamina class (2023).
URL https://puolustusvoimat.fi/en/equipment#/asset/view/id/331

[64] Finnish Navy, Hämeenmaa class (2023).
URL https://puolustusvoimat.fi/en/equipment#/asset/view/id/324

[65] K130 braunschweig class corvette (January 2008).
URL https://www.naval-technology.com/projects/k130corvette/?cf-view

[66] Ministry of Defence, Squadron 2020 - the finnish defence forces' strategic project (2017).
URL https://www.defmin.fi/files/3819/Squadron_2020_The_Finnish_Defence_Forces_strategic_project.pdf

[67] Finnish Navy, Pohjanmaa class (2023).
URL https://puolustusvoimat.fi/en/equipment#/asset/view/id/326

[68] P. Liu, A General Theory of Fluid Mechanics, Springer Singapore, 2021. doi:https://doi.org/10.1007/978-981-33-6660-2.

[69] L. S. Shapley, Stochastic games*, Proceedings of the National Academy of Sciences 39 (10) (1953) 1095–1100. arXiv:https://www.pnas.org/doi/pdf/10.1073/pnas.39.10.1095, doi:10.1073/pnas.39.10.1095.
URL https://www.pnas.org/doi/abs/10.1073/pnas.39.10.1095

[70] E. E. A. (EEA), Baltic sea physiography (depth distribution and main currents) (2012).
URL (https://www.eea.europa.eu/legal/copyright)

[71] J. D. Hunter, Matplotlib: A 2d graphics environment, Computing in Science & Engineering 9 (3) (2007) 90–95. doi:10.1109/MCSE.2007.55.

[72] P. E. Hart, N. J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE Transactions on Systems Science and Cybernetics 4 (2) (1968) 100–107. doi:10.1109/TSSC.1968.300136.

[73] C. Payne, Principles of Naval Weapon Systems, Bluejacket Series, Naval Institute Press, 2006.
URL https://books.google.fi/books?id=F3q59-hcGDoC

[74] Office of the Chairman of the Joint Chiefs of Staff, DOD dictionary of military and associated terminology (2021).

[75] A. J. Palmer, D. C. Baker, A novel simple semi-empirical model for the effective earth radius factor, IEEE transactions on broadcasting 52 (4) (2006) 557–565.

[76] F. G. Abdollah Ghasemi, Ali Abedi, Propagation Engineering in Radio Links Design, Springer, 2013.

[77] Furuno, Furuno Marine Radar DRS X-class series (2023).
URL https://www.furuno.com/special/en/radar/drs6ax-class/

[78] L.-F. Huang, C.-G. Liu, H.-G. Wang, Q.-L. Zhu, L.-J. Zhang, J. Han, Y.-S. Zhang, Q.-N. Wang, Experimental analysis of atmospheric ducts and navigation radar over-the-horizon detection, Remote Sensing 14 (11) (2022). doi:10.3390/rs14112588.
URL https://www.mdpi.com/2072-4292/14/11/2588

[79] L. Rautiainen, J. Tyynelä, M. Lensu, S. Siiriä, V. Vakkari, E. O'Connor, K. Hämäläinen, H. Lonka, K. Stenbäck, J. Koistinen, L. Laakso, Utö observatory for analysing atmospheric ducting events over baltic coastal and marine waters, Remote Sensing 15 (12) (2023). doi:10.3390/rs15122989.
URL https://www.mdpi.com/2072-4292/15/12/2989

[80] J. E. Bresenham, Algorithm for computer control of a digital plotter, IBM Systems Journal 4 (1) (1965) 25–30. doi:10.1147/sj.41.0025.

[81] J. C. Schulte, An analysis of the historical effectiveness of anti-ship cruise missiles in littoral warfare, Master's thesis, Naval Postgraduate School (1994).

[82] E. Rubinstein, Technology in war and peace, IEEE Spectrum 19 (10) (1982) 34–35. doi:10.1109/MSPEC.1982.6501945.

[83] F. Akar, Analysis of the sunken Russian cruiser Moskva and implications for Russia and the world navies, Horizon Insights (2022) 1.

[84] Merivoimat, Merivoimien uusi pintatorjuntaohjus - Gabriel 5 (December 2019).
URL https://merivoimat.fi/-/merivoimien-uusi-pintatorjuntaohjus-gabriel-5

[85] SAAB, The RBS15 family (2023).
URL https://www.saab.com/products/the-rbs15-family

[86] State Kyiv Design Bureau, LUCH, State Kyiv Design Bureau (November 2020).
URL https://www.luch.kiev.ua/images/data/en/LuchEn.pdf

[87] E. Wertheim, Hamina Class: Finland's modernized missile craft enhance baltic defense (October 2022).

[88] A. Ng, S. Russell, Algorithms for inverse reinforcement learning, ICML '00 Proceedings of the Seventeenth International Conference on Machine Learning (05 2000).

[89] M. J. Armstrong, A stochastic salvo model for naval surface combat, Operations Research 53 (5) (2005) 830–841. doi:10.1287/opre.1040.0195.
URL http://dx.doi.org/10.1287/opre.1040.0195

[90] J. Orr, A. Dutta, Multi-agent deep reinforcement learning for multi-robot applications: A survey, Sensors 23 (7) (2023). doi:10.3390/s23073625.
URL https://www.mdpi.com/1424-8220/23/7/3625

[91] R. Toro Icarte, E. Waldie, T. Klassen, R. Valenzano, M. Castro, S. McIlraith, Learning reward machines for partially observable reinforcement learning, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 32, Curran Associates, Inc., 2019, p. 15523–15534.
URL https://proceedings.neurips.cc/paper_files/paper/2019/file/532435c44bec236b471a47a88d63513d-Paper.pdf

[92] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y Arcas, Communication-efficient learning of deep networks from decentralized data (2023). arXiv:1602.05629.

[93] H. Nekoei, A. Badrinaaraayanan, A. Sinha, M. Amini, J. Rajendran, A. Mahajan, S. Chandar, Dealing with non-stationarity in decentralized cooperative multi-agent deep reinforcement learning via multi-timescale learning, in: S. Chandar, R. Pascanu, H. Sedghi, D. Precup (Eds.), Proceedings of The 2nd Conference on Lifelong Learning Agents, Vol. 232 of Proceedings of Machine Learning Research, PMLR, 2023, pp. 376–398.
URL https://proceedings.mlr.press/v232/nekoei23a.html

[94] H. van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double q-learning (2015). arXiv:1509.06461.

[95] L. Biewald, Experiment tracking with weights and biases, software available from wandb.com (2020).
URL https://www.wandb.com/

[96] J. Schulman, P. Moritz, S. Levine, M. Jordan, P. Abbeel, High-dimensional continuous control using generalized advantage estimation (2018). arXiv:1506.02438.

[97] H. van Hasselt, A. Guez, M. Hessel, V. Mnih, D. Silver, Learning values across many orders of magnitude (2016). arXiv:1602.07714.

[98] T. Schaul, J. Quan, I. Antonoglou, D. Silver, Prioritized experience replay (2016). arXiv:1511.05952.

[99] J. L. Ba, J. R. Kiros, G. E. Hinton, Layer normalization (2016). arXiv:1607.06450.

[100] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need (2023). arXiv:1706.03762.

[101] HEADQUARTERS OF THE COMMANDER IN CHIEF, Radar pickets and methods of combating suicide attacks off okinawa (1945).
URL https://www.history.navy.mil/research/library/online-reading-room/title-list-alphabetically/b/battle-experience-radar-pickets.html