Master's thesis 2021 Swapnil Kumar

NTNU
Norwegian University of
Science and Technology
Faculty of Engineering
Operations and Civil Engineering
Department of Ocean

Swapnil Kumar

Deep Reinforcement Learning based Energy Management in Marine Hybrid Vehicle

June 2021

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

Swapnil Kumar

Deep Reinforcement Learning based Energy Management in Marine Hybrid Vehicle

Master's in Product and Systems Design
Supervisors: Vilmar Æsøy
Co-Supervisors: André Listou Ellefsen

June 2021

Norwegian University of Science and Technology
Faculty of Product and Systems Design
Institutt for havromsoperasjoner og byggteknikk - IHB

NTNU
Innovation and Creativity

NTNU

Norwegian University of Science and Technology

Master's in Product and Systems Design

Faculty of Product and Systems Design
Institutt for havromsoperasjoner og byggteknikk - IHB

ISBN (printed version)
ISBN (electronic version)
ISSN 1503-8181

Master's thesis at NTNU,

Printed by NTNU-trykk

Summary

Typical Energy management strategies used in Hybrid marine power include rule based or reactive control techniques. The overall goal of this thesis is to explore Energy management & Control techniques for hybrid marine vessels based on Deep Reinforcement Learning(DRL).The Aim of this research is to see the potential of replacing current Energy management strategies with modern online adaptive techniques that do not need any training data and can learn from its own experience.

Firstly a simple Hybrid marine power plant has been studied. A decision process has then been devised using a Markov Decision model.The decision model has been divided into Environment,Agent , Reward and actions. Different Entities involved in the Energy management process of aforementioned system have been assigned these roles.

The aim of this model is to maximise fuel efficiency during operation by controlling the type of actions the system takes during an operation cycle. The choice of action in this case are the engine power levels.The overall aim is to optimise the sequence of action or policy that can handle the operational loads.

The decision process is supported by a DQN or Deep - Q Network that accesses the value of each action at a certain level of load and storage State of charge (SOC). To learn this behaviour the Agent is taught using $\epsilon - Greedy$ algorithm which allows the agent to sufficiently explore the Environment , while slowly learn to exploit learned behaviour from memory. A physical setup to employ the same in real life was also suggested.

Finally a simulation has been performed to compare the decision behaviour with systems with two different Energy storage Capacities. The results show that DRL can be used as a control method for policy optimisation in terms of reducing Specific Fuel Oil consumption(SFOC) during a cycle by efficiently using the Energy storage.

The DQN agent responsible for decision making was able to reduced the percentage of actions that lead to higher SFOC to less than 5% of the time.Even when the Storage Capacity was reduced the Agent was able to adapt towards similar behaviour. The Results of simulation show that the agent was able to maximise the usage of Energy Storage while making sure that there is always sufficient amount of charge left , to run the engine at a level which consumes fuel more efficiently (lowest SFOC).

The simulation show that the lowest average SFOC for a non hybrid setup containing a single Marine diesel engine is around 0.88 at its optimal running level .Similar SFOC is obtained while using non optimised random actions in a Hybrid setup. When using The DRL based control on an hybrid this value can be reduced to 0.6. Similar values are obtained using the same decision model even when the Energy storage capacity is reduced by 30%.

Acknowledgements

I would like to express my deepest gratitude to my supervisors: Prof. Vilmar Æsøy , and André Listou Ellefsen, at Department of Ocean operations and Civil Engineering, NTNU Ålesund .They provided much needed guidance and support throughout the course of this project and its an understatement to say that without their inputs and encouragements this research wouldn't have been possible.

I would also like to give my special thanks to Saumitra Dwivedi,PhD Candidate IKT, NTNU, Ålesund .He allowed me to use his workstation for running computationally intensive calculations and simulations. I would also like to thank him for the moral support at several occasion during the tenure of this research.

It would also be an understatement to say that this project would not have been possible without support of NTNU and its resources. I would like to offer my deepest thanks to NTNU for allowing me this great opportunity to work on such an Innovative and

In the end, I would like to thank my family and friends for all their support and motivation.

Preface

This masters thesis is a submitted as a part of Master of Science degree in Product and Systems Design program at Norwegian Institute of Science and Technology (NTNU), Department of Ocean Operation and Civil Engineering. This research work was carried out during the final semester (spring 2021) of the master program.

This thesis explores Reinforcement Learning Techniques with regards to Energy management in a Hybrid Marine Vessel. Several techniques have been investigated for their applicability in control of the Load distribution for a standard Hybrid Power Plant containing a single Diesel engine and an Energy storage Device.

Optimizing the Load distribution between the Engine and Energy storage, so as to improve the Vessel's fuel Economy is the main focus of this research. This can been done by running the engine in a range where it utilises the fuel with maximum efficiency while maximizing the efficacy of the Energy storage.

As a Product and Systems Designer , I recognise the potential of Reinforcement Learning, and strive to formulate its applicability in Control of Physical Systems.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter structures the motivation and background of the research domain and provides an overall understanding to the research objectives and scope.This chapter also contains an overview of the Problem in question, this question shall be the center of research in this project . The chapter is finally concluded with an Introduction to over all structure of the thesis with brief description of each entity in the structure.

1.1 Background and Motivation

Automatic control has been one of the highly discussed subjects in recent years. Industries such as the maritime , has readily endorsed Automation , and are moving swiftly towards a possible future where a fully functioning ship can be run without the intervention of humans.The Automation of such degree spans much more than just automatic navigation. Automation of other operation process like Power production and Energy management is equally important.In other words if the future of a fully automatic ship is to become a reality , not only the movement and its control should be automatic but also the process that fuels the movement and other necessary activities.

Maritime industry is known for its high efficiency, in terms of energy usage.It is responsible for over 80% of the global trade , yet contributes to only 9.26% of greenhouse gas emissions.The contribution of carbon emissions form the current Maritime vessels can , however can not be neglected as it accounts to over 1000 million tons of CO_2[66][18].The Concept of Hybridisation, however has paved way to better fuel Economy and the possibility of Emission Reduction.[46] When A hybrid Ship containing a Power source such as motors and an Energy

storage , are used efficiently large savings in fuel cost and emission reduction can be achieved. This can be done by running the Engine at a level where it uses the fuel most efficiently.The power produced by engine at this level however, might be less or more than what is demanded at any time. Hence the surplus of deficit is accommodated using the energy storage such as battery Sets or Super Capacitors.

Deciding the distribution of power between these sources has been a challenging task. Maximising the utilisation of fuel and Energy storage needs continuously changing policies or series of actions which are able to adapt to changing load demands. This has so far been done by using simple PID based controllers, Rule based algorithms along with other techniques[12] [61] .A These algorithms do offer good degree of optimisation , yet their efficiency varies greatly from One load Profile to another.In other words these controls are reactive and do not learn from a previous memory or experience[24].

Predictive control introduces the memory factor , which is able to learn from previous experiences and decide the best action to take at any moment.Prediction of load however can not be employed until done so in real time and may again vary for different cycles.[58] [41]. Reinforcement Learning based Energy management has been shown to be promising alternative ,mainly due its ability to adapt to different Load conditions. The 'on the go' learning approach offers flexibility to the control as it is independent from the need of previous data and can adapt and self tune parameters to maximise the desirable behaviour.[24]

The concept of Reinforcement learning is not new and has been successfully used in several Industry for adaptive control. The use of RL is most prominent in the Gaming industry where Trained RL agents have shown to be at par with or many times better than the best players in the world[33] [65]. It has also shown promising results in Fields like Finance [13] [35],Smart grid Management [80]and resource allocation[72].

The possibility of using RL to overcome the daunting task of maximising the potential of Resources of a hybrid power plant, makes it an exciting field to discover and study. In such cases control systems have to continuously work towards figuring out what , when and how much of a resource(engine or battery) to use to satisfy the load. Doing so while considering the fact that these resources also need to be used at a certain level to maximise their individual operational efficiency , while load itself is continuously changing in random ways , makes such task very difficult for humans.

With overall goal of maximum fuel efficiency such a concept also offers the possibility of financial benefit through fuel saving as well as lower maintenance of

engine, maximum use of Energy storage and lower emissions.

The motivation of this thesis is based on three aspects. First, the thesis contribute towards mapping the Energy management process of a hybrid ferry into a Decision problem .The relevancy of this lies in the fact that, this would enhance the readers' understanding of the of system , as well as shed some light on how methods such as reinforcement learning can be used to tackle such problems. Secondly the thesis aims to contribute towards discovering the use of Deep Reinforcement learning in Energy management with a Hybrid Ship a Case. Thirdly , the thesis strives to improve the current Energy management strategy.With the use of DRL the motivation of the thesis lies in improving overall operational efficiency in terms of better fuel usage and possible emission reduction through efficient use of fuel.

1.2 Problem Overview

The load profile of many marine vessels is highly stochastic and dynamic in nature. Allocating the load between an Engine and battery can be challenging task .As previously mentioned Deciding when , and how much power should be allocated to each of aforementioned becomes more challenging when one aims to maximize fuel economy.Other factors that should be considered while deciding this divide are, higher maintenance of engine at lower power levels, battery life efficient use of Energy Storage.

It has been previously proposed that a good amount of economic saving can be obtained using strategic loading with energy storage devices [43]. But the authors here also state that there is a need of deeper understanding of the engine nature at every possible operation condition. Peak-saving was used along with charging and discharging at a derived set points.

While Rule based algorithm are simple and reactive in nature , they do not guarantee same efficiency for different load profiles.[41] Other predictive Algorithms such as load prediction rely on previous data[3][74] and cant be applied until the model has been trained with lots of historical data. Such data is not only hard to find but are often unlabeled , besides the process itself is often offline.

While Reinforcement learning techniques such as Q learning , or Fuzzy Q learning offer the solution to this problem , they often suffer from the curse of Dimensionality[53]. Which basically means that the problem of optimal decision making becomes exponentially complex in systems with large number of decision variables.Making Q learning , and similar processes inefficient for systems with large number of decision variables.

Artificial neural networks are known for their efficacy of mapping complicated

functions such as the Value function (used in Q learning or similar processes), which is generally regarded an easier approach towards handling multi-variable decision making of larger dimensions.Combining them with Reinforcement learning techniques result in something called a DQN[1] -deep Q network.This solves the dimensional issue of the aforementioned.

To properly use such a tool , it is required to have explicit mathematical definition of the decision problem and to map the process of decision making. This in normally done through following the conventions of a MDP - Markov decision process , which divides the whole process in terms or Actions , Rewards , Environment and Agent. Together representing the dynamics of a decision process. [53]

Hence it is important to correctly translate the operation EMS control for a hybrid power plant into an MDP model so tools such as DQN can be used effectively.

[1]DRL - or deep reinforcement learning can be seen as method , where as DQN or Deep Q network can be seen as one of the 'tools' to employ DRL ,as more of a programming approach , in this project DQN and DRL has been used interchangeably.

1.3 Scope

The scope of this thesis lies within the following boundaries.

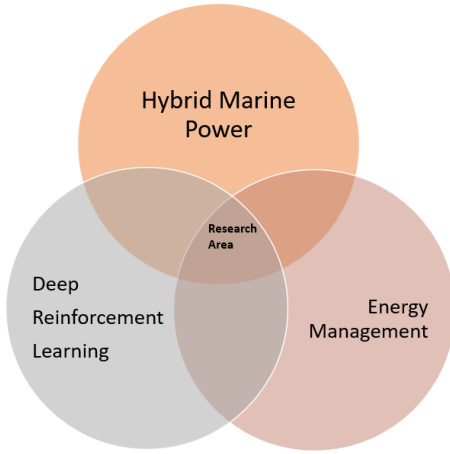


Figure 1.1: Scope of research

1. Marine hybrid power plant: The Operation of a Single Diesel engine and generalised energy storage as any possible combination of Battery and Super-capacitors or another energy storage device.[2]

2. DRL Based Online control- To achieve intelligent memory based real time control strategy for using combination of Reinforcement learning and Deep neural networks(DQN) for generating a sequence of possible actions or policy so as to maximize desirable system behaviour.[3]

3. Energy management System (EMS) Control : The thesis has its boundary in control of distribution strategy of load[4] between the Energy production unit such as a motor or engine and Energy storage unit by managing the deficit or surplus between them.[5]

[2]Design decisions have not been addressed in thesis.Moreover the thesis assumes no distinction between type of energy storage.The possibility of using any combination of a battery pack and Super-capacitors may add further intricacies such as weight and design challenges.

[3]In this thesis the desirable behaviour has been limited to minimizing Specific Fuel consumption and maximizing energy storage efficacy.

[4]Overall load considering all primary secondary and hotel loads in the ship , along with compensation of losses occurring in all mechanical and electrical transmission

[5]Charging and discharging cycle of storage as well as affect of the same on lifecycle are not a focus of this research. Although a brief description of working of some component of this control

1.4 Objectives

The process of learning through trying has been the part of human life ever since the beginning of human race.This is somewhat the essence of Reinforcement learning as well, just that instead of a human a machine or computer learns through trying. We have just begun to understand the power of machine learning and its an understatement to say that we have a long way to go.

"A failure is not always a mistake, it may simply be the best one can do under circumstance,the real failure is to not try."

- B.F Skinner

This thesis aims to solve a very small set of problems and present the findings as a contribution to the knowledge base in this domain.In support of the above mentioned scope, the following research questions were studied and addressed:

1. How to Represent the problem of EMS Control for marine Hybrid powerplant as a Decision making process or MDP?

In a MDP it is quintessential to properly represent the Physical system in an understandable mathematical model. This model when solved using methods such as Reinforcement learning outputs the desired policy.To test the model , tools such as programming can be used.This thesis intends to use a simple case to represent the above mentioned problem.

2.Why and How DRL be used for policy optimisation to reduce Specific Fuel consumption during any operation cycle?

The thesis discovers the applicability of DRL or DQN in solving the problem of Energy allocation between Energy storage unit and Power production unit. The aim of the thesis is to model an agent that learns to utilise knowledge previously gained to predict the best action and overtime learn to maximise the designed desire-able , in this case the reduction in specific fuel consumption.Am online approach has been presented allowing model to learn 'on the go'.

3.Which Deep Learning model yields better compatibility with the given model of Reinforcement Learning?

There are several possibilities to how a deep learning model can be setup in collaboration of a reinforcement learning algorithm,and also each of them have several

system has been presented, the overall design of the physical system is not the primary focus of this research.

tweak-able parameters which can significantly influence the performance of the overall model. This thesis aims to test some of deep learning models to check compatibility with the problem described above.

1.5 Thesis Structure

This thesis has been divided in to chapters . These chapters have been arranged to facilitated readers understanding in a systematic manner. A brief description of these chapters are given below.

Chapter 2 - Background Concepts: Describes relevant theory for this thesis. This includes theory about a Hybrid power plant used in maritime industry,The energy management system and control strategies used and Reinforcement learning and finally ,a brief description of how neural networks can be used in Reinforcement learning has been presented.

Chapter 3 - Related work: Explores research relevant to this thesis. The chapter gives a summary of EMS control with traditional methods, but also modern approaches for policy optimisation in EMS . An introduction to how these modern methods ,have possible advantage or disadvantage over the traditional methods have been summarised in this chapter using examples of recent research in this field.

Chapter 4- Methodology : Presents the methodology used in this thesis. This includes data,details of simulation practices, implementation details, and an overview of the cases studied in this thesis. These include 2 different Deep-learning models paired with double Deep Reinforcement learning algorithm.Each of these models are tested for two different cases.

Chapter 5-MDP model and Test Cases : This Chapter explains detail the formulation of the problem into an MDP model , and Solves the problem using two test cases.These cases vary in the sense that the Energy Storage capacity in the later case has been reduced to see how the decision model fares in this case.

Chapter 6-Discussion : This chapter discusses the results obtained from formulation of the mathematical model based on an MDP structure, their solution based on the solution strategies above mentioned in a comparative way.

Chapter 7-Conclusion: This chapter discusses the results obtained by answering the Research questions. It also describes the contribution from this thesis and states possible ideas for Future work.

Chapter 2

Background Concepts

This chapter discusses the Theories behind the main concepts used in this research. The chapter starts with a brief description about Hybrid ships in general , their history and development and how they have become an integral and emerging part of the maritime sector.This includes the impact of using hybrid concepts on the carbon emission and fuel saving in maritime sector.

The Chapter then goes deeper into the maritime hybrid concept by shedding some light on the working of a maritime motor and how output levels are related to fuel utilisation efficiency. The concept of specific fuel oil consumption(SFOC) and its relevancy to a maritime hybrid power plant is discussed.

The use of Energy storage device in Hybrid power is discussed later in this chapter with main focus on how fuel efficiency can be increased with proper usage of this resource.

Deeper into the chapter , the concept of an Energy management system and its relevancy to optimising a hybrid power plant setup has been introduced , as a new section. This section includes the details about some energy management strategies. Traditional energy management strategies have been discussed along with how these strategies have been implemented in the past.

This section goes deeper into the concept of energy management strategies by introducing rule based controls and newly developed machine learning based control, concluding the section.

A general introduction to reinforcement learning is given in a new section, the concepts and workings of RL has been discussed in brief with focus on relevancy to this research.

The concepts used in this project mainly include MDP and methods to solve an MDP problem such as, Use of Q-learning and Deep Q learning along with other reinforcement learning techniques. The use of Deep learning in relevance to solving an issue related to Deep Reinforcement learning has been addressed.

Since Deep Q learning is the main tool used in the project , it has been given a section of its own . Here the concept has been explained in detail. The relevant parts have been introduced as subsections , such as Double DQN , Deep learning models used in DQN , details about parts of a reinforcement learning network as well as some learning strategies.

The chapter concludes with this section explaining details of Exploitation vs Exploration dilemma faced by an agent.

2.1 Hybrid Power

Hybrid propulsion in this chapter refers to a combination of conventional propulsion and electrical propulsion. This chapter discusses the Environmental and other effects of Carbon Emission in maritime sector. Brief history of hybrid propulsion in general , Diesel (engine-generator) propulsion and concluding with Energy Storage system used in a hybrid propulsion.

2.1.1 Pollution in Maritime Industry and Environmental Regulations

Even the most energy efficient diesel engine engine , a large oil tanker emits more than $300,000$ tons of of CO_2 per year , equivalent to a medium size coal power-plant on land.And there are more than $93,000$ large ships in the world as 2017 [52]

This number has been increasing since the industrial production has shifted from USA , and Europe to China and other Asian countries[52]. This means the ships need to travel longer and farther. Since majority of the trade in the world takes place in the sea , and the fleet of ships responsible for this trade are majorly power through conventional carbon based fuel , its safe to assume that the carbon footprint from the maritime sector has increased many-folds in recent years.The UN estimated an increase of 30% in CO_2 emission from 2012 to 2020. The problem of emission however is not limited to carbon footprint , pollutants such as Sulfur and Soot emissions.The implications of health, especially respiratory problems can also be a major issue in Countries bordering the High-marine traffic areas.[52]

MEPC - Marine Environment protection committee in April 2018 reaffirmed its commitment to reducing GHG - Green House gas emission from International shipping. With an Aim to mitigate 50% of GHG emissions by 2050 ,as compared to 2008 ,one of the primary focus has been put on increasing Energy efficiency.

[26] [27] [28] Emission reduction approaches with respect to Alternative / New fuel include[71]:

- Electricity
- Fuel Cells
- Wind
- LNG
- Ammonia
- Sustainable biofuels
- Solar....

While operation practices to reduce carbon footprint from maritime industry include:

- Speed optimisation
- Ship port interface
- Onshore power
- Ship Size

In response to the Paris Agreement in 2015, the IMO adopted an Initial Strategy for reducing GHGs caused by ships in April 2018. As shown in Figure below, this Initial Strategy is to reduce the total annual GHG emissionsby 50% by 2050 compared to 2008, and aims to decrease the CI(Carbon Intensity) by 40% by 2030, and by 70% by 2050 to de-carbonize as soon as possible within this century. In order to establish a joint response strategy for the IMO regulation of GHGs, a variety of programs with the industry involved have been developed and carried out along with various research activities in between member states.[30]

It can clearly be seen from the above mentioned guidelines proposed by IMO and MEPC , that there is an immediate need to deviate from the current reliance on fossil fuels , if reduction in carbon and other pollution levels is to to be achieved. This makes concept such as Hybridisation an essential part of future of maritime industry as it has potential to significantly increase the energy efficiency.While renewable energy alternatives such as wind and solar are in infancy when it comes to

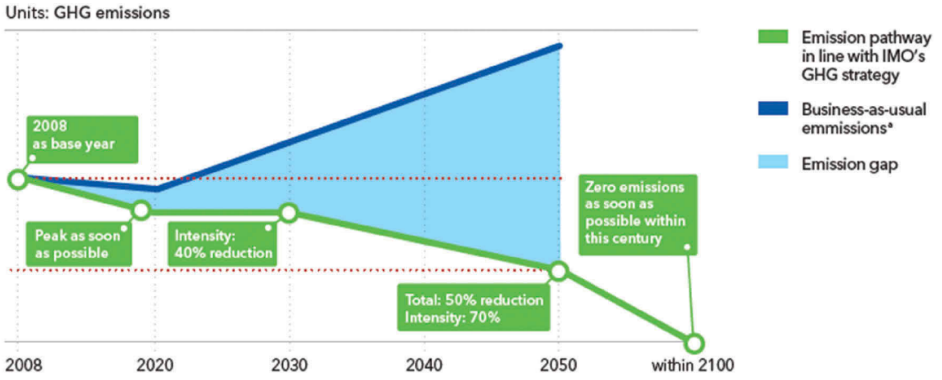


Figure 2.1: GHG emission gap between IMO GHG strategy and BAU emissions (DNV-GL 2019).

their usage in maritime industry , greener fuel alternatives such as LNG , Hydrogen and Fuel cells have not not matured in terms of technological feasibility.

It can be said that ,although total abandonment of current fossil fuel based propulsion system in maritime , is not feasible , but getting most out of the fuel , to reduce wastage and control carbon footprint is the next best possible option , at-least until the aforementioned greener fuel alternatives have been developed sufficiently.

2.1.2 Hybrid Power

Hybrid power generally refers to a combination of conventional power with electrical power. In maritime systems hybrid propulsion is fueled by a generator running on fuel , and supported by an energy storage device. The most commonly used generators use MDO - Marine diesel oil or HFO- Heavy Fuel Oil . The Energy storage , usually is in form of a battery set . While Hybrid propulsion is not a new concept , its application has been limited and only a minor portion of the world fleet runs on Hybrid propulsion.As of November 2020, this number is around 250.[4]

The ship propulsion system with mechanical drive and reduction gears was first developed in the United Kingdom. Following the development of the first large elec- tric motor and generator in 1910, electric propulsion for ships was developed in the United States and elsewhere.Electrical propulsion however did not become part of the main stream maritime propulsion technology until later 1980's. Although used in submarines due to unavailability of air needed for running engine underwater , the first commercially used electrical propulsion came into being in

only few cruise ships such an *Canberra* in 1960 and *Normandie* in 1936.[52]

The development of Electrical propulsion , opened the doors to using motors to drive propellers instead of directly attaching propellers to the engine drive shaft shaft.This also allowed the propulsion to be powered using another electrical power source like a battery. Taking advantage of this capability , a Hybrid propulsion was developed.

A common layout of a hybrid propulsion looks like the following [21].

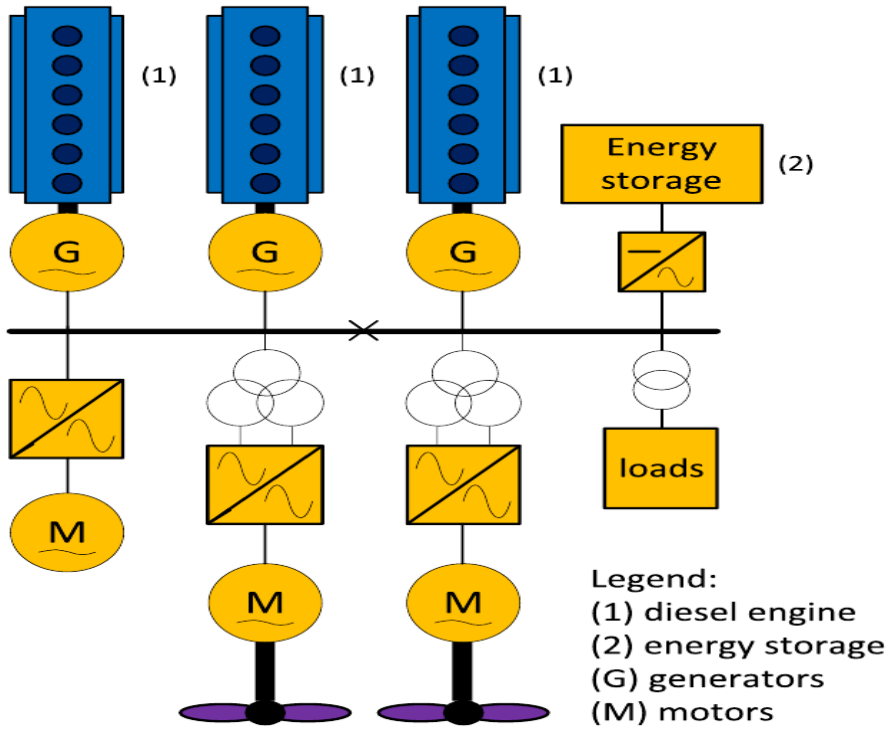


Figure 2.2: Typical Hybrid Hybrid propulsion System with hybrid power Supply

As shown in the picture 2.2 . A hybrid propulsion usually uses electrical motor , this motor supplies to the propulsion and other power output needs and Loads. These motors are usually connected to transformers and converters that convert Electrical power from generators and Energy storage into usable power for the motors. While Generators produce power , when driven by typically a diesel engine , the energy storage, in normal circumstances , receive power from generators, store it in form of energy, and give necessary power for aforementioned motors.

A hybrid propulsion system has shown to reduce fuel consumption and emissions up to 10–35%, while improving noise, maintainability, manoeuvrability and comfort[21] .Generally , the primary power producing units , i.e the diesel engine have higher efficiency at a certain range. When used in this range , they tend to be most economical. In a Hybrid Propulsion system , the energy storage allows the Engine to run at this range , by supplying or absorbing the surplus between the Load demand and the power produced by the Engine.

In this research , this advantage of a hybrid propulsion has been studied.The focus of this project lies in ,identifying the benefits of using advanced intelligent control to determine the power distribution between the Diesel Engine and the Energy storage to maximise the advantages of an hybrid Propulsion system.

2.1.3 Marine diesel Engine

To maximise the benefit of using a hybrid propulsion, it is necessary to understand the working of marine Engines.As previously mentioned these Engines have a tendency to have higher efficiency in a certain range.When this range is known , the control of an intelligent system can be directed towards this range.

The average diesel engine has a thermal efficiency of around 30-40 % . In other words only about one third of the heat energy contained in the fuel is being converted to useful power.The rest is lost in form of heat and other losses.Although this sounds wasteful ,these engines are considered to be more energy efficient than Gasoline engines which tend to have an efficiency around 25-35 % [10]

While overall thermal efficiency of deals in overall output of an engine , it does not specify the efficiency at various load levels. Since all marine Diesel engines are capable of handling varied loads , it is necessary to understand its operation cycle and associated fuel economy to develop control to maximise the fuel utilisation potential. A good way to do so is monitoring SOFC - Specific Fuel Oil Consumption , which denotes to the the amount of fuel used to produce one unit of power. Previously done Experimental and Simulation studies have shown that generally every engine has a range where the SOFC is minimum. In other words it can be said that , when running these engines in a certain range the engine extracts more energy as compared to other ranges.Simulation studies indicates this range to be around 80% load level, for a two stroke low speed diesel engine This study is shown below.[83]

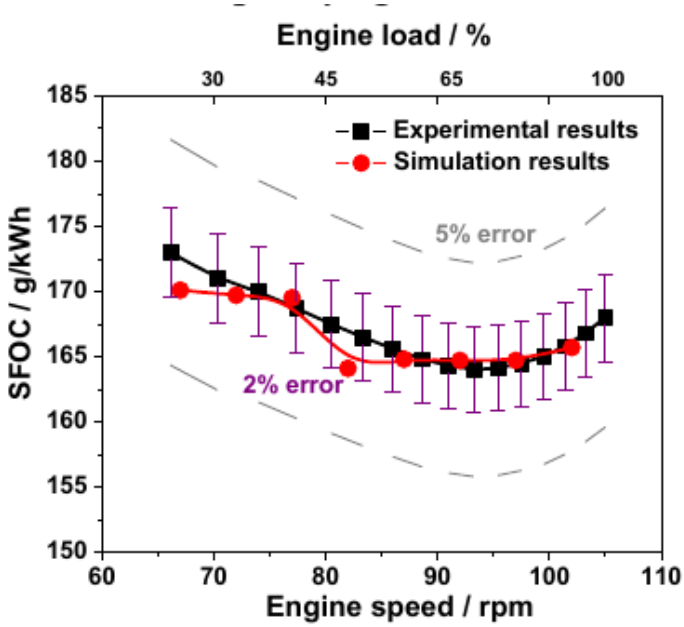


Figure 2.3: Specific Fuel Oil Consumption (SOFC) - Marine diesel Engine [81]

Similar studies have been proposed , using different analysing techniques , yet yielding the same result.[82] This project uses the fuel efficiency model from a two stroke marine diesel engine produced by John Deere , model -4045TFM50 capable of producing 57 -71 Kw of power @ 1500 -1800 rpm. This engine is supplemented by a turbo system. The fuel characteristics of this model are given in the the table below.[29]

Table 2.1: Fuel Consumption data - John Deere 4045TFM50 (ref nogva

Effect and Consumption	**Speed (O/min)/Hz**			
Speed (O/min)/Hz	**1500/50**		**1800/60**	
Motor Effect	57 kW		71 kW	
power level 100%	14.4 L/t	215g/kWh	17.6 L/t	211g/kWh
power level 75%	10.6 L/t	211g/kWh	13.4 L/t	214g/kWh
power level 50%	7.5 L/t	224g/kWh	9.7 L/t	232g/kWh
power level 25%	4.3 L/t	256g/kWh	5.6 L/t	268g/kWh

It can be seen in 2.3 and 4.1 , the SOFC at tends to be higher at lower load levels , as the engine load is increased , the SOFC increases and peaks around 80% . After that the SOFC again tends to increase but usually is lower than that at lower load levels. In other words these engines can utilise maximum out of the fuel at around 80% power level. Hence it can be safely assumed that running the engine in this level will maximise the utilisation of fuel.Even though the fuel consumption is higher at this stage the overall produced output power per unit weight of fuel is higher.

The reason behind this kind of fuel efficiency characteristic is beyond the scope of this research.However this characteristic shall be used later in the project to design ideal operation control of the hybrid propulsion system , in terms of control of engine.

2.1.4 Energy Storage

Energy storage is an integral part of a hybrid propulsion system.It allows the engine to run at a certain power level , and allows Independence from the load by taking care of the surplus between the load and engine power.

There are several kinds of energy storage available , but the ones we would concentrate in this research can be divided into two parts:

1. Electro-chemical storage
2. Capacitance based storage.

Electrochemical storage involve array of cells which are able to produce electrical potential using chemical reactions.These cells usually are connected to in series or parallel to increase capacity. There are different types of electrochemical storage available ,each has its own set of advantage and disadvantages. some of them have been presented in the table below.[85]

Comparisons of battery technologies.

Battery type	Advantages	Disadvantages
Lead-acid	√Low cost √Low self-discharge (2–5%per month)	× Short cycle life (1200–1800 cycles) × Cycle life affected by depth of charge × Low energy density (about 40 Wh/kg)
Nickel-based	√Can be fully charged (3000 cycles) √Higher energy density (50–80 Wh/kg)	× High cost, 10 times of lead acid battery × High self-discharge (10% per month)
Lithium-ion	√High energy density (80–190 Wh/kg) √Very high efficiency 90–100% √Low self-discharge (1–3% per month)	× Very high cost ($900–1300/kW h) × Life cycle severely shorten by deep discharge × Require special overcharge protection circuit
Sodium Sulphur (NaS)	√High efficiency 85–92% √High energy density (100 Wh/kg) √No degradation fo deep charge √No self-discharge	× Be heated in stand-by mode at 325 °C
Flow battery	√Independent energy and power ratings √Long service life (10,000 cycles) √No degradation for deep charge √Negligible self-discharge	× Medium energy density (40–70 Wh/kg)

Figure 2.4: Types of Electrochemical battery

One of the popular choices is the Lithium -Ion battery ,due to above mentioned advantages.However these batteries have a limited battery life which is highly dependent on the charging and discharging cycle. Generally the battery life (measured in cycles) reduces if the battery is discharged below a certain level.While lead - acid battery has shown to have drastically reduced battery life if discharged below 50% , while recent research on lithium ion batteries have shown to have increased battery life if cycled between 0-50 % power levels.Charging and discharging at varied rates have also shown to greatly affect the battery life. [85].Due to this the operation control does not take into account the battery life.

The other kind of energy storage is the Capacitance based energy storage. This type of storage does not use any kind of chemical reaction for electrical potential generation. The capacitance based storage allows charging and discharging at a varied rate , also the life of these kind of units are expected to be quite large.

Capacitance based energy storage is comparatively newer concept , however its potentials are being recognised and in near future it can very well be a feasible alternative to conventional battery packs. However the current form of super- capacitors suffer from a much lower energy density as compared to a conventional battery storage.Which means that a smaller battery pack can store similar amount of energy as compared to a much larger super capacitor. This limits the applicability of super capacitor to a great extent.

Some of the advantages and disadvantages of Capacitance based Energy storage is given in the below.

1. Advantages
    (a) Long cycle life > 100000 cycles
    (b) Good power density under certain conditions , limited by IR or equivalent series resistance (esr) ,complexity of equivalent circuit.
    (c) Simple principle and mode of construction (can employ battery construction technology)
    (d) Cheap materials (for aqueous embodiment)
    (e) Can be combined with battery for hybrid applications
2. Disadvantages
    (a) Limited energy density
    (b) Poor volume energy Density
    (c) Low working voltages (compared with electrolytes , decent as compared to battery)
    (d) require expensive materials such as pure water free materials for non-aqueous solutions
    (e) Good matching of cell units is necessary

However , the field of capacitance based energy storage has seen some great strides in terms of research and development and the future of this technology seems to be quite bright.

A comparative study between capacitance based energy storage and Conventional energy storage is given in.[16]

This thesis does not specify the kind of energy storage. The only characteristic considered in designing the control system is the Energy capacity.However understanding the characteristics of the storage system helps us understand the need for this generalisation. Also keeping in mind the high pace industrial development around this technology , this research has been done keeping in mind a general benefit of energy storage in hybrid system , rather than a specific technology.

2.2 EMS- Energy management System

EMS or Energy management System in this Research refers to the energy distribution system of the Marine vessel . In a Hybrid setup there is usually a combination of Primary Power source i.e. An Engine and An Energy storage Device . The EMS enables the transfer of energy between the Source and Storage device. Usually the

primary source consist of a mechanical output device which need to be converted into more usable electrical power. For this purposes a generator is used.

Depending on the number of generators connected to the primary sources , There is a need of conversion of the generated electrical power into feasible and use-able characteristics such as phase,DC or AC type current, Voltage and Current. The EMS connected to the Converters enables this operations as one or more primary sources can be coupled with an Energy storage Device in a proper way.The overall load is then supported by giving a combination of power from the primary load and the Energy from the Storage device. In other words it can be said the EMS acts as a junction which enables interaction of power from various sources in the order to fulfil the load demands.

Discussion on the intricacies of the Power management system however, is beyond the scope of this research and only the operational aspects of the energy management system , such as the policy based on which it works , has been discussed.

2.2.1 Energy management Strategies

The use of a primary energy source and an Energy storage Device allows various combination of the two to fulfil the loads. These loads usually are highly stochastic in nature. The aim of the strategy is to always fulfil the loads 100% , ensuring unhindered operation throughout.

While ensuring the total fulfilment of the loads , it becomes a challenging task to design a the strategy for the energy management to ensure optimal working , as each load , and choice of actions ,significantly alters the state of the energy system.

Strategies such as ECMS[49] [60] introduces a Cost function to the electric power supplied by the Energy storage ,by considering the Battery as a reversible fuel tank. More advanced systems decide the distribution of power between the energy storage and primary source using some form of a PI controllers , including fuzzy PID. these shall be discussed in the next chapter. ¨ The use of Rule based approach in the formulation of EMS strategy has also been shown, this enables the controllers to decide what decisions to take on the basis of some Rules. Some of these rules are formulated on the basis of the guidelines given by entities such DNV -Gl

These rules are primarily based on expert opinions or in-house simulation of the system.For example the ships based on hybrid systems need to keep a minimum level of battery storage at a particular time of operation.

Rule based strategies such as refered in [32] , uses a set of IF and ELSE rules based on the current state of the system to determine which action to take. These rules are primarily a representation of the state and corresponding approach towards what

the author of the rules perceives as an optimal state of the system. Often these rules are based on Expert opinion.

The MPC or the model predictive control is regarded as the State of the art in terms of the Control of Hybrid Energy systems. These control methods use mathematical models to obtain future states. With predictive models such as MPC various strategies can be applied to the models , for testing and validation.Some of the common strategies are [43]:

- Enhanced dynamic performance: It is known that generator loading should be gradually ramped up. Increasing the load too quickly might lead to a blackout. The ESD can supply energy to the power plant during large load steps, and the generator will be loaded gradually. This will improve the safety and robustness of the system. In operations where large loads are expected, such as drilling, the vessels can operate with a smaller back up power supply because the ESD will compensate for abrupt power surges.
- Peak shaving: The generator-set power supply should be bounded between a lower and a higher limit, and the generator-set load variation should not exceed a predefined magnitude. This operation is important in cases where engines might automatically start and stop, such as during DP operations, leading to reduced efficiency in case of excessive engine running. Peak-shaving reduces fuel
- Energy Reserve :Recent developments in class rules and governmental regulations allow an ESD, with certain requirements, to act as a spinning reserve. Hence, for redundancy purposes, fewer generators need to be connected to the bus at any point in time. This can be used to move the load per generator toward the optimal working condition and thereby reduce the fuel consumption and emissions.
- Strategic Loading :By charging and discharging the ESD, it is possible to strategically load the generator. Through high/low engine load cycles, it is possible to lower the average fuel consumption and emissions compared to a system without strategic loading. The viability of strategic loading is directly related to the engine fuel consumption curve characteristics. Strategic loading viability requires a study for each individual vessel.
- Zero Emission Operation:By shutting down the generators and using ESDs only, it is possible to operate without any emissions. A large ESD is required to supply the power demand from the vessel. This operation is interesting and may become a requirement in the future for operations in ports and harbors. It is also the sole operational mode on fully electric vessels.

2.2.2 Machine Learning in EMS Control

New approaches with load and MPC have been developed based on ML as well . These models use training data from previous experience to device a model that predicts the forthcoming Loads. Such models suggest charging and discharging strategy. New models including Fuzzy algorithms have also been proposed for control. Although these models are able to predict Loads based on a trained data set , its efficiency is limited to similar load profiles only.

Since these models use supervised form of machine learning , they require extensive data and training varies according to availability and accuracy of training data sets.

While supervised form of learning requires extensive data set for training , validation and testing , reinforcement learning is based on 'Learning on the Fly'. This means that the model itself learns from its past experience. RL thus is helpful in cases which do not have a large amount of data recorded.

2.3 Dynamic programming : Decision Model Setup

The optimisation of a problem over time arises in several cases ranging from Inventory management ,distribution of donated organs for ones in need , or simply playing tic-tac-toe. Usually the problems involve a sequence of observations , making decisions , making more observation and so on , this process is known as a sequential decision making process. These sequential decision making processes are usually very easy to formulate but very hard to solve for an optimal sequence. [53] While methodically , fields like engineering and economics deal with continuous state space and control and decision process of the systems in these fields are aptly named as '*Control*' and are addressed under the umbrella term of Control Theory. Models which are built with a discreet space are modelled under the MDP process.

There is often a similarity, of stochastic , in dynamic models irrespective of the nature of state space.This leads to high amount of uncertainty to the decision making process.

Although dealing with an Engineering control project we shall be addressing the problem in form of an MDP. This shall enable us to use a discreet action and state space, which in-turn will facilitate computing later in the Project.

Simple modelling framework for any dynamic programming consists, minimum of the following:[54]

1. The state variable : This is the encapsulation of the information that we

need to make a decision. This also includes the information of how the state evolves over time.

2. The Decision Variables: This is the representation of the control process.
3. The transition Function : This signifies the effect of the Decision variable on the state.
4. Exogenous Data : This is the set of Data that becomes known at the Starting of every time step.
5. Contribution function : The Reward or Punishment the Decision maker receives as a result of the action and its consequences.
6. The objective function : What we want to achieve by the end of the whole Sequence.

Since we will be addressing the problem of Control from a sequence point of view , the understanding of the process of decision making will more also be sequential . We will Define the problem in a dynamic way and provide the above mentioned information. This hopefully will allow us to use a method that uses adaptive learning to make decisions in a sequential manner.

2.3.1 Sequential Decision Making and Markov's Decision Process- MDP

In this subsection we shall discuss the formulation of a sequential decision making process , through a widely accepted mathematical tool called a MDP - or Markov decision process. This section will also lay the foundation to the understanding the concept of reinforcement learning, explain its significance in sequential decision making.This will also help the reader understand the analogy and structure of a sequential Decision making process which backbones a Reinforcement learning process.

In a Decision making process such as MDP , there is a decision maker which is called an Agent.This Agent Interacts with the environment that its placed in. These interactions occur sequentially over time. The agent is able to perceive the environment at each time step. This perception is then used by the agent to make the next interaction with the environment , called an action. Now with each interaction ,the Environment transitions into a New state and the agent receives a reward as a consequence of its previous action.The sequence of action through out the whole decision process is called a Policy. Hence the Components of an MDP can be summarised as follows.[51]

- Agent

- Reward
- Environment
- Action

These components are associated with each other in form of a feedback loop. The diagram below shows the interaction. The deliverable from a MDP process are

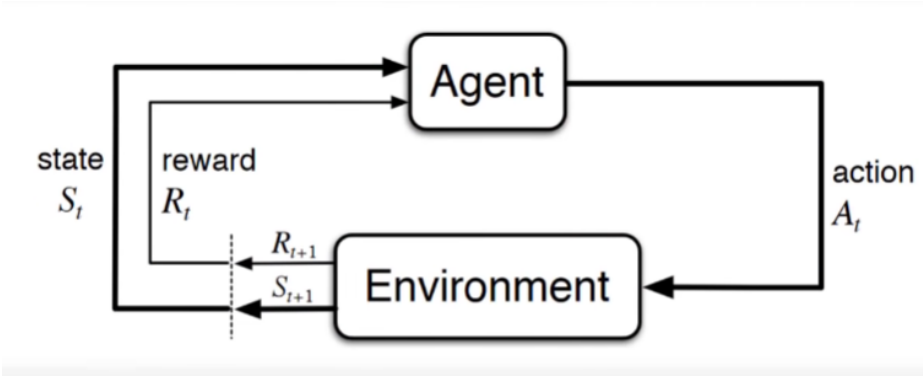


Figure 2.5: Decision Loop in An MDP

Q table representing the 'quality' of all possible action- state pairs ,as well as a Policy which is nothing but a series of actions for the whole process. The over all quality of the whole decision process is determined by the quality of the Policy. This quality is usually determined by combining the *cumulative* rewards obtained throughout the policy.[57]

A brief understanding of these concepts and their relevance to this project ,this is hereby discussed in brief. These concepts have been applied to the problem later in the report.(ref sec problem formulation)

In this Thesis , the Dynamic model of the Control system has been modelled on the basis of the Notations rules provided in [54].

In most sequential decision making process , we have to focus on finding a policy or sequence of actions , that maximises a certain optimisation problem. Depending on the type of problem , several optimisation methods including the one used in this project , can be used. Several of these methods have varying range of applicability towards different problems and its necessary to understand what is the nature of these problems and why a certain method is applicable to this problem.

Categorically the MDP problems can be classified into two groups :

1. Finite Horizon Problems : These kind of of problems have a very specific Horizon , or a finite number of steps over which the sum of the optimised value can be found.For example a game of Tic-Tac-Toe or any other game which terminates over time.

2. Infinite Horizon Problems : These problems do not tend to have a defined termination point and has to be optimised for all time. For example Showing ads in a browser , which adapts to the user preference , which might evolve or change over time for an infinite amount of time or certainly an undefined amount of time.

There may also be a case where a finite Horizon problem does not terminate , For example a game of chess where the participants are allowed infinite number of moves. One can find a case where the game does not terminate in such case. [7]

Both of these cases have different methods which have been used to solve an optimality equation for that specific problem. Some of the popular methods of solving an MDP for the above mentioned type of problems have been discussed in the section below.

2.3.2 Methods for solving an MDP

Based on the type of Horizon problem the solution methods for an MDP varies greatly in terms of computational complexity , for example in a finite horizon problem its just a matter of starting with the final time step and compute backwards the value of each possible state and choose the best out of these state sequences, which brings us to our first and simplest solution techniques.

Note: All the algorithms in this project use standard MDP notation style defined in Chapter 5.1 Notation style : for MDP design. [55].

Step 0. Initialization:

Initialize the terminal contribution $V_T(S_T)$.

Set $t = T - 1$.

Step 1. Calculate:

$$V_t(S_t) = \max_{x_t} \left\{ C_t(S_t, x_t) + \gamma \sum_{s' \in \mathcal{S}} \mathbb{P}(s'|S_t, x_t) V_{t+1}(s') \right\}$$

for all $S_t \in \mathcal{S}$.

Step 2. If $t > 0$, decrement t and return to step 1. Else, stop.

Figure 2.6: Backward Dynamic programming

One of the most used algorithms for an infinite horizon problem is the Value Iteration .It involves estimating the value function iteratively. At each iteration the estimate of the value function determines which decisions we will make and as a result we define the policy.

Step 0. Initialization:

Set $v^0(s) = 0 \ \forall s \in \mathcal{S}$.

Fix a tolerance parameter $\epsilon > 0$.

Set $n = 1$.

Step 1. For each $s \in \mathcal{S}$ compute:

$$v^n(s) \quad = \quad \max_{x \in \mathcal{X}} \left(C(s,x) + \gamma \sum_{s' \in \mathcal{S}} \mathbb{P}(s'|s,x) v^{n-1}(s') \right). \tag{3.22}$$

Let x^n be the decision vector that solves equation (3.22).

Step 2. If $||v^n - v^{n-1}|| < \epsilon(1-\gamma)/2\gamma$, let π^ϵ be the resulting policy that solves (3.22), and let $v^\epsilon = v^n$ and stop; else set $n = n + 1$ and go to step 1.

Figure 2.7: Value iteration algorithm for infinite horizon optimisation

There are variations of Value iteration method such as the Gauss- Siedel variation , Relative Value iterations etc. One other important aspect of VI is the low estimate of initial iterations. This defines the bounds and rates of convergence of the VI procedures.[6]

The other very popular method involves the policy iteration . In which we choose a policy and then find the horizon, discontinued value of the the policy.This value is then used to choose a new policy.In its simplest form the algorithm looks something like.[55]

Step 0. Initialization:

Step 0a. Select a policy π^0.

Step 0b. Set $n = 1$.

Step 1. Given a policy π^{n-1}:

Step 1a. Compute the one-step transition matrix $P^{\pi^{n-1}}$.

Step 1b Compute the contribution vector $c^{\pi^{n-1}}$ where the element for state s is given by $c^{\pi^{n-1}}(s) = C(s, X^{\pi^{n-1}})$.

Step 2. Let $v^{\pi,n}$ be the solution to

$$(I - \gamma P^{\pi^{n-1}})v = c^{\pi^{n-1}}.$$

Step 3. Find a policy π^n defined by

$$x^n(s) = \arg\max_{x \in \mathcal{X}} \left(C(x) + \gamma P^{\pi} v^n\right).$$

This requires that we compute an action for each state s.

Step 4. If $x^n(s) = x^{n-1}(s)$ for all states s, then set $x^* = x^n$; otherwise, set $n = n + 1$ and go to step 1.

Figure 2.8: Policy Iteration

This method has been modified to accommodate various kinds of value iterations,such asmentioned in [36] Along with the above mentioned techniques there are also hybrid -value policies as well as linear programming formulations available to solve an Infinite horizon problem. However these will not be discussed here. Our focus in this course revolves around Reinforcement learning. It was introduced previously that the reinforcement learning models such as Deep -Q-Learning , do not rely on previous data. Now we shall discuss the topic in detail.

2.3.3 Reinforcement Learning

Reinforcement learning is a model free approach to develop understanding of a model. This type of approach is used to understand the dynamics when a proper mathematical model is not available. This is often used to predict the behaviour of a system when the co relations between composing entities is too hard to formulate.This is also true for any exogenous Information related to a model, specially When randomness of an Information relevant to a model behaviour is too high.

Secondly the transition between the states may be unpredictable. When we are not sure , which state the system would transition to given a certain initial state ,action and information. Irrespective of how accurately we may be able to map the discrepancies of state transitions , there is always a big chance that the model generated by that mapping may not be a true model.

Data that come from a physical process has a virtue of having incomplete information. For example we may not be able to know the set of outcomes , or the probability of the outcomes of a model. Thus the 'model free' approach directs us to an approach where we may not need a one step transition matrix , but where we may have a transition function. Solving problems this way is usually a part of the umbrella term - model free approximate dynamic programming.

In other words Reinforcement learning is the mapping of situations into actions and learning what to do from experience.In this way the learner is not told what to do , like in other forms of machine learning. but instead is allowed to discover on its own what is the best type of action for any situation.In a way this resonates highly with the human way of learning things.Reinforcement learning is comparatively new , as compared to other supervised learning process. [69]

Although Supervised form of learning is an important aspect of machine learning ,it is highly dependent on the set of data it has been given for its training.This however only works for recognising patterns which are comparatively hard to change , and depict a very specific behaviour or feature. The question arises that will these supervised models work , if there is a significant change in the behaviour , or feature of the object? For example if a ML model has been trained to identify a certain type of fruit , say apple , will it be able to recognise a misshaped apple? or will it be able to distinguish between a pear which has a shape of an apple? The applicability of these models thus can be questioned in uncharted territories. This is where Reinforcement learning comes into play.

A number of different algorithms can be applied to Reinforcement learning. Some of these algorithms are direct , like Q- Learning . The policy iteration in reinforcement learning is applied through algorithms like Actor-Critic models. The difference between the two models is that Actor - Critic method does not use a replay buffer , or previous memory to actively learn, it rather relies on a critic.In this thesis we shall discuss the Q learning approach towards Reinforcement learning.

2.4 Q-learning : First Step towards reinforcement Learning

To understand Q learning a brief understanding of TD or temporal Difference learning is necessary. Temporal difference can be deemed as one of the core concepts of reinforcement learning. This is generally viewed as a combination of monte carlo methods and dynamic programming ideas. It is similar to the Monte Carlo methods in a way that it can learn directly from the raw experience without the model of the environments dynamics. Its similarity to DP can be tracked to the fact that TD estimates based in part on other learned estimate , without waiting for the final outcome , i.e bootstrapping. [77]

Since TD methods do not require a model of the environment , they have a certain degree of advantage over DP methods. TD can be applied in an online fully incremental fashion thus giving it the advantage over the Monte-Carlo methods.

The use of TD in policy iteration algorithms for solving MDP can be given in form of SARSA TD- Control . This method follows the usual pattern of generalised policy iteration (GPI) as the name suggests. However in this thesis we have limited the focus to an off-policy algorithm known as Q- Learning. [70] Q- Learing as mentioned earlier is an Off-policy TD control algorithm. It was developed by Watkins in 1989 . In its simplest form Q-Learning is defined by [75] [53]

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max Q(s_{t+1}, a) - Q(s_t, a_t) \right] \tag{2.1}$$

here Q is the learned Action-value function directly estimates Q^* , which is the optimal value of the action value function, independent of the policy being followed.This dramatically simplifies the analysis of the algorithm and enables early convergence proofs.

The Q learning algorithm can be described as shown below

Algorithm 1: Q learning Algorithm

1 Initialise $Q(s_t, a_t)$ arbitrarily

2 Repeat (for each episode):
Initialise **s**
Repeat (for each step of episode):
Choose **a** from **s** using policy derived from **Q** (likeϵ-greedy)
Take action **a**,observe **r,s'**
$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max Q(s_{t+1}, a) - Q(s_t, a_t) \right]$
$s \leftarrow s'$
until **s** is terminal

This shall be the initial framework and concept for the design of this project. However this algorithm does come with some disadvantages in terms of the size do the state space , as we will see in the next section sections.

2.4.1 Deep Q Learning

While simple Q learning solves the problem of reiteration in the full set of data for solving a MDP , like in a VI method it still has disadvantages of tits own.When the set of observable state is very large the method tends to be computationally less efficient. For example lets take an example of a state space with 1000 $\times$1000 state

space and 10 actions , which gives us the a Q table with 10 million Q values .Some systems such as the cartpole may give us an extremely large set of these values. This is popularly known as the *The curse of Dimensionality*

DL models have been known to be a good way of approximating functions, and q value is too a function , hence it can be approximated using a DL model.

DQN in its most simple incarnation can be called a fusion between a Q learning technique , with a DL model , here this model is used to replace the gigantic Q table , specially for Large State space.

With some simple modification a Q learning algorithm can be converted into a DQL algorithm .[37]

Algorithm 2: Deep - Q learning Algorithm

1 Initialise $Q(s, a)$ with some initial approximation.

2 By interacting with the environment obtain the Tuple$(s, a, r, s^{'})$.

3 Calculate loss $\mathcal{L}$ = $(Q(s, a) - r)^2$ if the episode has ended or
$\mathcal{L}$ = $\left(Q(s, a) - (r + \gamma \max a' \in A)Q_{s',a'}\right)^2$

4 Update $Q(s, a)$ using **Stochastic gradient Decent** SDG by minimising the loss with respect to the model parameters.

5 Repeat from Step 2 until converged.

We shall be suing this algorithm in this Project.Although this algorithm looks simple enough , its applicability is questionable , as many things can go wrong.

Some of them have been discussed below which will give us a way to modify this algorithm to avoid these discrepancies.

2.4.2 DQN- Deep Q network

[1] This sub-section explains the pitfalls of using the above model for control of a System. It addresses issues that are related to the model and suggests relevant solution to the problem. Finally a modified algorithm is presented.

[1]This subsection is from Simulation point of of view ,the reader can decide to skip this if details from the implementation point of view is not necessary

The interaction with the environment

When using DL to face reinforcement learning , we need a certain amount of data to train on .This can be done by taking random decisions and checking how good it is for an action i.e exploration. However if do so in a sequential way , we may end up waiting a very long time before and two feasible sequence of action take place let alone optimal.

As an alternative we can also use the Q table to approximate a source for the behaviour.Now the question arises , that what if the source , i.e the Q table is not good ? Which in-turn is bound to happen in the beginning of the training. Is 'exploitation' of the Q values a good option?

With a little though it can be interpreted that if Q table is bad , the agent will be stuck with taking bad decisions only , without trying to take a better action. This is popularly known as **Exploration VS Exploitation** Dilemma. On one hand Our agent needs to build a complete picture of the transitions and action outcomes , on the other hand it is not optimal to wait for a very long time before we can have a decent sequence of actions. We should also be able to learn from the experience , while not entirely relying on it.

Common sense tells us that taking random actions in the beginning of an episode seems to be a good way to go , as Q value representation in the beginning is bound to be bad. But as we do this more and more we start to get an initial understanding of the dynamics governing the transitions and action behaviours.We should be able to utilise this understanding to choose a near optimal series of action for the elapsed set of observations.

A popular way to do so is $\epsilon - Greedy$ Algorithm.This algorithm mixes the two extreme behaviours together. When this algorithm is used to take an action the agent switches between random and Max Q Value select , behaviour based on a Hyper-parameter known as ϵ. This basically helps us select the ratio of random actions . Say if ϵ was 1 ,(100%) the probability of taking random actions can be set 1. The usual practice being to start the training with maximum random probability and gradually lower this as the training proceeds. This is also known as $\epsilon - decay$.

This solves the Interaction and Model interaction Dilemma which was previously mentioned.

SGD optimisation

One of the key components of our model is somewhat based on prediction of a complex non , linear function Q(s,a). We have decided to use a NN to do this prediction. IT can be said that one of the core of Q learning procedure is borrowed

from supervised learning itself. Here we use bellman equation to calculate the target for this function. But a couple of question arises, taking into account one of the fundamentals of SGD optimisations , concerning the training data , that is that the training data should be **Independent and Identically distributed**. Which surely doesn't happen to be the case.

1. Firstly the data that we use for training , our samples , will be very close to each other and more or less dependent on each other.
2. Secondly , Distribution of our sample is dependent on the policy that we choose for taking actions . Here we have decided to use the $\epsilon - Greedy$ to take actions , hence the data generated in our sample shall also be distributed according to this algorithm.

To counter this challenge we introduce a *'Buffer'* known as **Replay**. What Replay does is that it stores previous experience in a memory , this memory is of fixed size and prioritizes new results , to save and ejects older experiences as the new ones come in. We might then randomly chose a single sample or a mini-batch and use it for the training. This method helps us to get more or less an independent data.This however is the simplest method of sampling , other more sophisticated methods exists as well as mentioned in [38]

Correlation between Steps

The issue not having an training data to sample from , once again causes an issue with the model DQN Algorithm. This however arises due to the fact that the bellman equation provides us Q' to train Q , however there is only a single step involved between them , which will make very hard for the NN to distinguish between them.This has been compared to chasing ones own tail by the author [37]. It is like saying that a teacher asks a student to learn till page 6 of a book for a test , and by the time the she reaches that page , teacher decides that she will take test till page 11.

To solve this issue we use this trick called a **target network** ,instead of 1 DL model we shall use 2 , which will identical to our original but uses bellman to generate Q'. The weights of this model is copied periodically to the original model . This period is usually large and like 1K or 10K time steps.

The Markov Property

Our model has been structured around an MDP . And a fundamental requirement of an MDP is that the Environment should be distinguishable from one another.Meaning that the data from observing the environment is all we should need

to make a decision.It should be made sure that the Agent has enough information available at every time-step which allows it to distinguish between states and in-turn take better decisions.

This is usually done by giving the NN the information about some kind of gradient between two consecutive states

In our case the environment contains the battery level denoted by the SOC and a Load which is randomly distributed along a certain time-period.Hence the question then arises that, does this query apply in our case. if it does how much can we benefit from allowing the NN to learn the gradient between any two consecutive states.

This is going to be one of the experiments that we perform in this research.

Final form of DQN training

After addressing the concerns mentioned in this section above , we have to now implement these changes in our algorithm. This will allow us to approach the problem relatively better. [37]

Algorithm 3: Final DQN-model

1 Initialise the parameters for $Q(s,a)$ and $\hat{Q(s,a)}$ with random weights , $\epsilon \leftarrow 1.0$, and empty the replay buffer.

2 With probability ϵ select a random action a,otherwise select a=arg $\max_{\hat{a} \in A} Q(s,a)$

3 execute action a in a emulator , and observe the reward ,r, and the next state, $s^{'}$

4 Store the transition (s,a,r,s') in the replay buffer.

5 Sample a random batch of transitions from the buffer.

6 For every transition in the buffer, calculate target y =r if the episode has ended at this step or $y = r + \gamma \max \hat{Q}(s', a')$ otherwise

7 Calculate loss $\mathcal{L} = (Q(s,a) - y)^2$

8 Update Q(s,a) using SGD , by minimising the loss with respect to the model parameters.

9 Every N steps copy weights from $Q$to $\hat{Q}$

10 Repeat from step 2 until converged

This algorithm[2] shall be used in this project, however some modification and additions shall be done to suit the case.

[2]This algorithm has been designed keeping video - games such as ATARI in mind

Chapter 3

Related Work

This Chapter discusses the works related to control of EMS.The chapter begins with introduction to tradition approaches towards Energy management Control in various systems followed by the relevant literature on Modern Energy management approaches. The chapter concludes with insight on how maritime industry can benefit from implementing modern Energy management control methods.

The systems in focus are Hybrid marine power , HEVs , Smart grids and Automation (mainly in robotics). The relevant research topics and popular keywords used for searches are Energy/Power management,Hybridisation , EMS ,Control , DL,DQN and MDP.

It is interesting to notice the sheer difference between the amount of research done about use of ML & RL in control of video-games and physical system control problems , specially maritime.We shall discuss in this chapter how Control concerning EMS in maritime is dominated by model based approaches and how AI and ML related research for control in other sectors such as HEV and smart grids have proven to be advantageous over these model based approaches.This certainly paves a promising path for doing the same for maritime industry as well.

Popular journals for these kind of research are MDPI ,IEEE Power and Energy Technology Systems Journal, and Journal of Control Engineering Practice.

3.1 Energy Management using traditional Techniques

We shall discuss energy management strategy as a concept at first , firstly classifying them on the basis of the approach , modern or radiation , while classifying them on the basis of the sector of technology they have been used in. We shall first discuss the traditional approaches , they include Every approach which does not include ML.

3.1.1 EMS - HEV

The existing energy management strategy is classified into three categories.[78] [22]

1. Rule based EMS
2. Load following strategy and
3. Electric assist strategy

While these strategies have been primarily used in HEV , Even in the maritime sector the strategies follow the same trend. These strategies were introduced in section 2.2.1

In HEV EMS , the strategies such as the ones mentioned in the list above rely heavily on the results of expensive experiments , extensive trial and error and rely very much on human expertise without having a prior knowledge of the driving conditions. [47].It is safe to mention that , Human experience based control can be prone towards personal bias , and may or may not yield the desirable control.

Control strategies which are a bit more advanced and use some form of PID control , employ heuristic control techniques such as fuzzy rules [39] [17]. These rules are easy to implement and have proven to be quite effective in the past , but questions have been arisen as for their optimal behaviour in different driving conditions.They have also been seen to be less flexible and hence critically limited by working conditions. Such algorithms consequently are non adaptive.

Another EMS strategy include the EMS - based on Optimisation . They take in the knowledge of previous driving cycle and predict the future driving cycles. These methods include Dynamic Programming (DP) [3] SQP [59] , GA [15] and PMP [79]. To perform optimisation. These algorithms have shown to be able to get optimal power distribution in hybrid systems , however this power split is only relevant to that driving cycle , in reality such methods are neither optimal nor charge sustaining. Also there is no way to apply these strategies directly hence are only relevant to offline application. AS we have discussed previously optimisation

strategies such as predictive , suffer from this thing called a curse of dimensionality , that makes most cases computationally impossible due to large state spaces.

Another type of optimisation is based on MPC[9] - model predictive control.In these methods a finite domain , optimisation control problem is solved at every sampling instance and the policy is optimised based on a rolling optimisation method.These methods have proven to be effective in the past and are known for their robustness and good control effect.

As someone really intelligent once said -

'All models are wrong just some models are useful'

The efficiency of these models are highly dependent of several limiting assumptions , mathematical accuracy and tedious mathematical modelling. [32]

3.1.2 EMS - Maritime

An example of how EMS in Maritime works is given below.

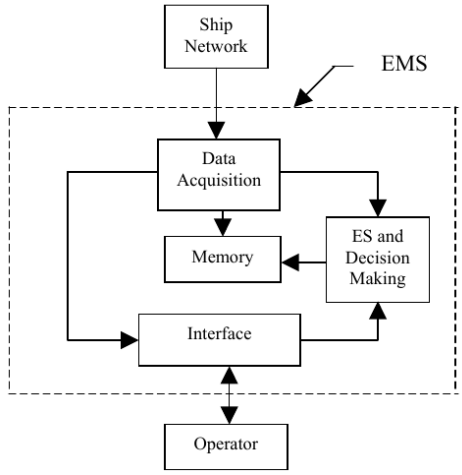


Figure 3.1: Block Diagram Representation Of EMS , (ES = Expert System

Since we are interested in implementing the ES part of 3.1 , we shall first the traditional strategies of control in maritime. Although not all of these strategies deal with control of the ES - Decision(ESD) making system , we shall pay attention to the relevant research so as to facilitate Implementation later in the project.

In a tradition hybrid power producer (engine +generator), the most common ESD

usage are

- Enhanced Dynamic performance
- Peak Shaving[23]
- Spinning reserve
- Strategic Loading
- Zero Emission operation

The ESD - Generator interaction is usually controlled using the ES system mentioned in 3.1. Maritime industry have in past been dominated by traditional EMS , which either use Expert opinion[63] , or Rules based on Expert opinion. These rules can , and most probably will suffer from bias towards ones specific field of expertise. Simple model based and Rule based approaches have more or less been replaced by more advanced approaches.

A typical EMS , is a DP - Dynamic program approach. This firstly evaluates other algorithms and second to analyse vehicle control laws,extract implantable rules in control, and improve rule based methods.However such algorithms like the ones mentioned in previous subsection are sensitive to driving cycles.

3.2 Energy Management using advanced approaches

With development in NN and AI , more intelligent approaches to control have been proposed.

3.2.1 Advanced EMS

The most initial approach to control using a predictive algorithm include , load prediction.[73] This is a learning based approach that utilises patterns and sequences learned from historical data.This has the advantage of online applicability.[84] One of the researches propose use of this technique for online control , using smart real-time communication and intelligent infrastructure.[42] This method considers vehicles as non -connected smaller grids which share data among each other.The issue with these methods are that they need complex control and Expertise to perform with any desirable efficiency. Moreover these methods are not something we can call END-TO-END.

The first time DRL was used for control of any EMS based HEV was for an PHEV in form of a fixed route plugin type hybrid bus.[86] While being able to apply DRL the method failed to integrate online learning, and was limited to the fixed route it

was designed for.Other critical issues related to this type of methods include the effect of immediate reward on the network , stability of the Q network among others. This issue was addressed and solution using a similar network was proposed in another research.This was able to make the process online.[58]

The sheer need of extensive previous data , makes such approach problematic. The use of Reinforcement learning solves this problem , however this process also is not very applicable where the state space is large , which makes computations very heavy.

Although these approaches have been seen to have some problems , they are promising in the sense that they are able to map non-linear operations of complex systems.

Online learning control strategies such as FQL , and NDP have been proposed for online control of HEV. These approaches do not rely on previous data , however these approaches require design of complex fuzzy controllers and previous expertise.[2] [68][14] [34]. Other use a more advanced load predictive model for the control. [44] [48]

Another very interesting field where Advanced control techniques have been used or at-least actively studied is the Smart -Grid.The problem of control in smart grid is very complex and wide. Some topics in decision making consist of , smart - home design , and automation, system integration of renewables , load demand modelling and PHEV.

Control in smart home management is quite an interesting area of research. Some of interesting control concepts in this field include - Zig-bee control [50] and multi agent system design[67] framework with control and balance between conflicting objective. These models are primarily designed for information gathering and transmission.They fail to clarify how , real time control can be implemented.

Other researches include , designing a virtual power plant[56] for control , and Integrating customer decision in Grid control[40].Although these researches do not directly relate to our topic of research , The problem of decision making in Smart grids are based on MDP hence in lieu of lack of research in EMS -for hybrid marine power plant , specially concerning a storage device , and a random load distribution is similar to a renewable mode smart grid control for an infinite horizon problem.

As per the knowledge of the author the problem of EMS control using a DQN has rarely been addressed in a published research , it is important to mention the two relevant research which address somewhat similar problems. First research includes EMS for HEV using a DQN [24] , and the other is EMS based on Q

learning autonomous control of auxiliary power network.[25] .These researches have been used as inspiration for this research.

It is also very important to address the issue of control in Automation, Robotic arm control and other fields. These researches show that DQL can be used for solving complex control problems with large state spaces.

Previously research has been done in the field of robotics [8], building HVAC[76] ,ramp metering[5] and other fields such as lane keeping [62] and autonomous breaking system [11] .Some more advanced model based behaviour prediction has also been done in terms of understanding the dynamics of these systems , they include the use of SINDY [31] and Koopman operators[1] for online learning.

While it is important to address the fact that most of the research concerning DRL is based on video games such as Atari [64] . They are functionally different from this problem in the sense that they deal with a virtual state space and often employ a convolution network for perception of visual inputs in form of frames , or screenshots of the games. These are problems of their own This project however does not face such problems as the State and action spaces are straightforward and can be represented in the dorm of simple mathematical sets .The understanding of these algorithms used in control of video games will help us solve the intricacies and difficulties we are bound to face while adopting a model free , policy free approach while designing control for this project.

Chapter 4

Methodology

So far the problem addressed in this project has been addressed, why is that a problem and how we plan to tackle it has also been introduced . We also introduced some related work which can be considered as state of the art in this specific field , or relate-able fields including Control in general , DQN based control of physical systems and traditional and modern techniques to control that can be used in EMS.

The main objective of this project is to define a control strategy for EMS in a simple Hybrid Marine power plant. Keeping that in mind we shall now introduced the methodology used in this report , The details about data , and other implementation details will be introduced.The methods used in this chapter shall be used to design the experiments setup and simulation in the following chapter. while section 4.1 is relevant towards experiment design and physical control parameters, section 4.2 is relevant towards the simulation of DQN network and the decision part of the experiment. Finally an implementation setup has been presented which will allow collaboration of the above two parts in a real life scenario. This will be a setup used for implementing Online control using the suggested setup.

4.1 Hybrid Power Plant

Since A major part of this project deals in formulating a strategy for the EMS.

The inspiration for EMS control was somewhat based on the Hybrid marine Diesel Engine setup in Hybrid Energy Lab of NTNU in Ålesund. Details of which has been presented in this subsection.

This setup consists of a marine diesel engine connected to a battery setup. This engine is calibrated with various sensors that measures different performance as-

pects of the Engine. However we are only interested in the fuel economy of the engine and shall not be bothering with the internals of the working of the engine.

The engine is connected to a battery setup , and a controller. The controller is a simple manual Input setup that allows the user to send in manual commands. This controller is unable to take frequent inputs ,say from a PID or Audrino specially regarding the power distribution between the Engine and the battery.

Keeping this in mind It was decided that a Simple simulation Can be done for checking the efficiency of the control process developed in the Project ,which shall allow us to focus on the Dynamics of the control rather than physical aspects and implantation.

Later in this chapter a method of introducing a physical control using advanced intelligent network will be introduced for physical implementations.

4.1.1 Engine Power

In this Project we are primarily concerned with the fuel Economy of a hybrid setup. The Engine being the only consumer of the Fuel is thus the source of this data. In a typical Hybrid power system ,A marine Diesel engine used fuel to produce power , usually inform of rotation. This rotational power is then transferred to a generator which converts it into electrical power. Our focus Ends at the Engine itself.

These Marine diesel Engine have various thermal efficiency at various speeds. They also have a different SOFC at different power levels of output. WE shall consider the working of an Engine in terms of Power level VS SOFC.

The setup we shall use for this purpose is same as the one placed in the hybrid power lab at the Department of Ocean Operations and Civil Engineering at the Norwegian University of Science and Technology in Aalesund. . This is a marine diesel engine is manufactured by JOHN-DEERE and the model number is 4045TFM50.

The relevant data was extracted from the catalogue of the Engine and is presented in the following table:

Table 4.1: SFOC of Engine : John Deere 4045TFM50

Effect and Consumption	**Speed (O/min)/Hz**			
Speed (O/min)/Hz	**1500/50**		**1800/60**	
Motor Effect	57 kW		71 kW	
power level 100%	14.4 L/t	215g/kWh	17.6 L/t	211g/kWh
power level 75%	10.6 L/t	211g/kWh	13.4 L/t	214g/kWh
power level 50%	7.5 L/t	224g/kWh	9.7 L/t	232g/kWh
power level 25%	4.3 L/t	256g/kWh	5.6 L/t	268g/kWh

As the table shows the engine power levels (at both speeds) of 75% and 100% have the lowest SFOC. This means that the engine needs least amount of fuel in this range **per KW of produced output. In**

We shall concern our-selves with this data only ,Further research focusing on predictive maintenance[20] [19] has also been done using ML, which only makes this research much more relevant in a sense of increased use ML in control of Physical systems.

4.1.2 Energy Storage

In this project we shall concern our self with a generic virtual storage device. Since previous research shows that batteries have limitations based on life -SOC ,and newer energy storage like super-capacitors are in their infancy, it was decided to consider Energy storage as an Ideal Case.

By this we mean that the the power storage can be charged and discharged at varying rate and the duration of charging or discharging is taken to be negligible. In this respect the device resembles a super-capacitor more.

This also means that the Energy storage shall be guided by the following equations only.

$$Storage_t^{Power} = \left(load_t - Engine_t^{output}\right) \tag{4.1}$$

and the change in SOC is described by:

$$\delta SOC_{\Delta t}^{change} = \frac{\delta Storage_t^{Energy} \times \Delta t}{Storage^{max}} \tag{4.2}$$

here storage $Storage_t^{Power}$ signifies the power taken from the storage or given to

the storage at any time instance. $load_t - Engine_t^{output}$ signifies the difference between the load and Engine Output powers.

For any time interval Δt the change in SOC is given by the formula 4.2. $Storage^{max}$ is the MAX energy storage capacity of the Storage device.

4.1.3 Load

Load profile used for this project was designed as random variation of a simple cycle consisting of loading , cruising , stationary , port charging and idle discharging cycles. [19] 4.1

The real life operation data has been introduced with varying randomness of up to 30% at every time step. Moreover since control is to be simulated in this project the time step has been chosen as 1 minute instead of 1 second , however to simulate randomness in cyclic load behaviour every cycle of load would vary from previous cycle and each episode shall consist of 3 episode. The load thus shall look something like this.

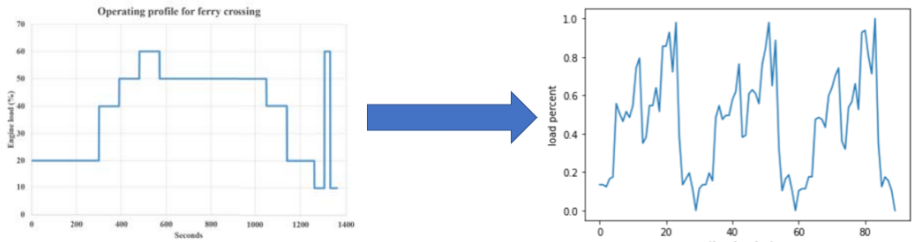


Figure 4.1: Modification of load to introduce randomness

In figure 4.1Left- Operating profile for a simulated autonomous ferry crossing. Right - Modified Load profile , upto 30% randomness at each time step , This load changes randomly for every cycle and every episode.

The Thesis centers around Independence from historical data and it would be hypocritical to use such data shown in left side 4.1.Also lack of openly available operational data makes simulation based on real life scenario difficult. Hence It was decided to used several variations of randomly generated data to test the efficiency of the EMS strategy.

4.2 Implementation Detail

4.2.1 Hardware

The experiments in this project was performed in combination of virtual machine available for general student usage in Google Colab. Some of the experiments were performed on a Dell Latitude 7490 laptop. with Intel (R) UHD Graphics 620 with 8GB of memory. The CPU in the laptop was Intel (R) Core(TM) i7-8650U CPU with 8 cores 2.1 GHz . The laptop has 16 GB of RAM.

4.2.2 Simulator

A simple simulator was created , this simulator simply converts the Input engine power into change in SOC. The structure this simulator is given below :

```
def get_next_state(current_load,current_soc,action_index):
    if actions[action_index]=='eng_0':
        soc_change=(0*max_eng_power-current_load*80)/(60*max_bat)
    elif actions[action_index]=='eng_25':
        soc_change=(.25*max_eng_power-current_load*80)/(60*max_bat)
    elif actions[action_index]=='eng_50':
        soc_change=(.25*max_eng_power-current_load*80)/(60*max_bat)
    elif actions[action_index]=='eng_75':
        soc_change=(.75*max_eng_power-current_load*80)/(60*max_bat)
    else:
        soc_change=(.95*max_eng_power-current_load*80)/(60*max_bat)
    new_soc=current_soc+soc_change
    return new_soc
```

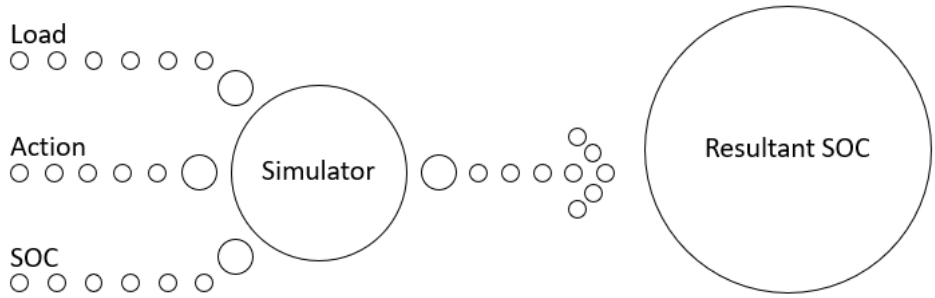


Figure 4.2: Simulator

The current load and Current SOC are fed into the Simulator along with the action Index.

The action Index is defined as index of the action corresponding to the list

```
actions=['eng_0','eng_25','eng_50',
'eng_75','eng_100']
```

As their names suggest they are engine load levels that the simulator will assume as the input and base the simulation on it and give the resultant SOC, which is the next state of the Energy storage system.

In this Project we shall consider only the static load levels. Dynamic loads have not been considered

4.2.3 Programming Language & Libraries

All development was done with the programming language python 3. Python is known for its easy and user friendly implementation style and is one of the most commonly used programming languages used in ML. It supports many libraries of ML and DL . This simplifies the job of creating ML and DL projects as we don not need to implement these concepts on our own. Most of the python code has great interoperatbility with low level C code , which allows good speed in well developed ML and DL libraries which are optimised for speed and accuracy. The main libraries used in this project are:

- Numpy : This library is used for analysis , exploration , modification and calculations performed for tabular data. It has many useful function which allow different mathematical operations
- Random: To generate randomness in data and operations.
- matplotlib :Used For plotting and data visualisation
- Tensorflow : For machine learning operations.
- collections : Data storage and mining.
- itertools: Iteration of Data during simulations

The next subsection describes the common implementation details and data formatting, normalisation, data splitting , DQL models and hyper parameters.

4.2.4 Normalisation of data and Formatting

The external data which we shall be working on is basically just load data , it has been assumed that the maximum load that the system can output is equal to the maximum engine load. The load is formatted in the form of time series data. The data is divided into 3 cycles of 30 minutes. Each data point has a value of

load between 75 and 0 KW. This cycle has been repeated with random variations multiplied by the data points to create variations of a single cycle. Several such cycles have been used.

The other data that we are concerned about is the SOC data and the action data. The controller has choice of 5 actions as defined in the previous section. These actions when fed through the simulator generates a change in SOC of the storage device.

Normalisation

ML benefits from data that works in similar scale. Usually this scale is chosen to be between -1 and 1. Since a major part of network used in this project is ML we need to normalise all the data mentioned above.We shall be using a simple MIN-MAX algorithm to normalise our data. This algorithm is defined below:

$$z = \frac{X - \min(x)}{\max(x) - min(x)} \tag{4.3}$$

since our data is not particularly dependent on the quantity, use of this algorithm should be sufficient .

4.2.5 Training: Deep - Q Learning

A vanilla dqn network was introduced in the previous chapter. We will now modify this network to fit our model.

The primary level of the DQN for EMS control shall contain the following parts. Note that this is the network we shall first use for training and then we shall test it on different load profiles

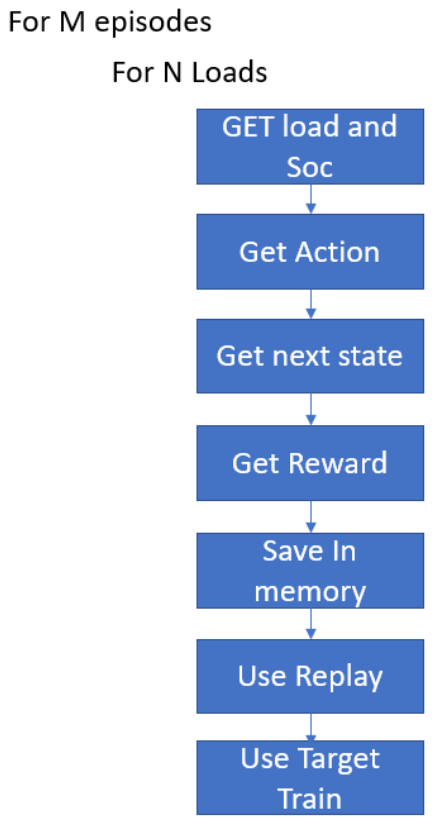


Figure 4.3: The DQN Network

Each of these Functions are explained below.

Get Load and Get SOC

When Each load becomes available - we need to normalise it to a value between 0 and 1. Also we should make sure we are working with the load associated with

that particular time period only.This is the job of the function Get Load.

Now in the beginning of the cycle we are not aware of the current state of charge of the battery. Hence it is logical to start every episode with a random soc. This shall be done by simply generating a random number between 0 and 1 , representing the percent of charge.

Get Action

Using Epsilon greedy algorithm we shall chose the action based on the DL network. We do this by using Get action function . What this function does is that it generates a random number and checks if that number is less than the value of epsilon. If the value of epsilon is greater than this generated action , it simply chooses a random action. Else , it inputs the value of generated SOC and Load into the network and chooses the best possible action from the DL network. In the beginning when the network has not been optimised this action is bound to be wrong.

Get next state

Now that the action has been chosen it is possible for us to get the consequent state. This is done by using the get next state function. The work of this function is to use current state (Soc) and the Action index to achieve the next SOC. This is done by using the simulator described in previous chapter.

Get Reward

The value of action and the next state are straight forward representation of the Engine use and the resultant soc.Hence its only relevant to use this value to get a reward.The work of this function is to get a reward based on aforementioned values. This reward shall represent the desired behavior in the state and action space. and should be normalised between 0 and 1.

Save in Memory

To use the replay function we first need to save the values of current state, next state , action taken and the reward achieved in a memory. This is done using the Save in memory function.This function stores these values in a limited storage. when this storage is filled after a certain number of episodes. The older values are pushed out to make room for the new values.

Replay Netowrk

Replay network is where most of the training of the DL network takes place. This is done using the values stored in the memory. This function activated when the

memory reaches a certain size. This is usually the size of the batch which we want to sample from. Once this size is reached a random batch of samples is taken from the memory, each value of current state is taken and a target value for the SOC is calculated using the target network.This target value represents the best action for that set of current state. Once the index of the optimal action is known the future state value for the best possible action is calculated using the bellman's equation. The target value is then modified by making its value equal to the sum of the current reward and the discounted value of the optimal future reward.

When all the values in the sample are modified through the Q network the original model is trained using the obtained values.

The training of DRL differs from simple DL , in a way that the data required for DRL is generated on the go , though a network called replay network. To use this network we need to first store the experience in a memory.After sufficient data has been stored in the storage , the replay network is activated. The following algorithm shows a general replay network [45]:

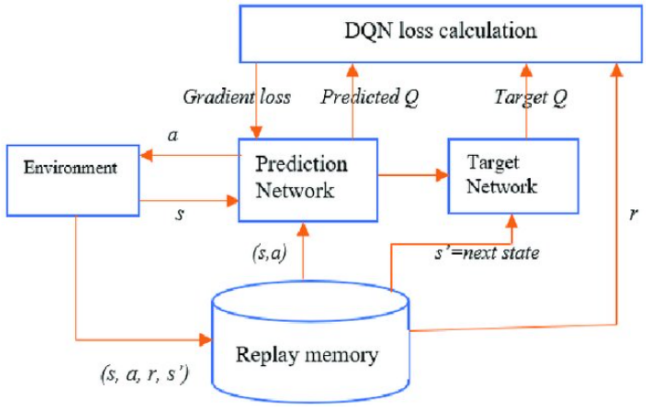


Figure 4.4: Replay network

The function of replay network is select a random batch of saved values from experience memory and apply RL algorithm on the target network.

Target Train

We discussed earlier that to achieve convolution we shall have to transfer the weights from the main model to the target model after a certain amount of training. This is done through the train target function. This function is activated every 500 episodes.

4.2.6 Hyper-parameters in DQN

Hyper parameters are values that determine some functions through the iterations of a network. In a DL based network such the DQN used in this project. These hyper-parameters guide the iteration process by fixing some of the variables used in each sub-function of the DQN. We shall now discuss what are these parameters and what do they do.

- Gamma γ : Discount rate used for calculation of Q value
- Epsilon ϵ: Defines randomness in choice of actions , according to ϵ- Greedy algorithm
- Tau τ : Used to discount current weight into target weights
- NN Hyperparameters (Dropout , Optimiser , Activation functions, Loss etc)

We shall not go in details of these hyper-parameters as this project simply aims to present the proof of concept , of applicability of DQN in control of EMS . In future however these parameters need to be studied in detail and further investigated or optimised.

4.2.7 Storage

In the network there are some storage used to facilitate the training , testing and validation of the network These storage re defined below:

- reward monitor :
- fuel-consumption-monitor :
- soc -memory
- action sequence
- reward sequence

These storage are used to store data for visualising results .

- memory : Stores the value of current state , action , reward and next state. This is the memory of the network from which the agent shall learn.
- train-list : This is a sub set of the above list used in replay function for training of NN

- memory list : Used for random sampling from the memory. Its a subset of the memory whose size is equal to the batch size.

They are used to create training data for the DL model for value function approximation using replay function.

4.2.8 Suggested Implementation

The physical implementation of the network in a real life scenario shall take place based on the following diagram.

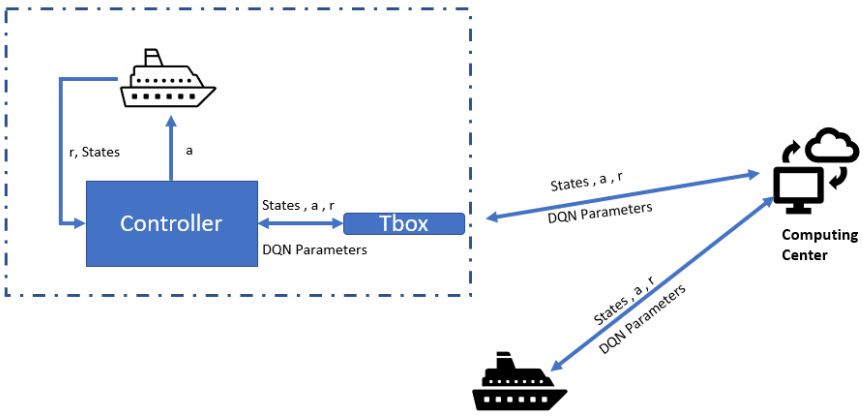


Figure 4.5: DQN Communication

1. There is an on-board controller which contains a DQN neural network and selects an action based on prediction from this network while storing the state - action transition
2. Another on board computer trains the network based on these learned values. This also sends necessary data in usable form directly to a communication center where similar data from various ships may be used to train a main learner network.
3. The on-board computer or remote computing center then obtains the state-action transitions, rewards and other necessary details from the action controller and trains the ob board network model.
4. The DQN is updated periodically by copying weights to the controller.
5. Communication between Controller and Computer is via Ethernet or CAN Bus.

6. If a remote computing center performs the training of th DQN network a vehicle terminal named Telematics-box (T-box) should be installed. This shall be responsible for transferring regular data for online training of the model.

This setup shall be responsible for communication of information between various entities. Using this setup an online control of EMS of hybrid vehicles can be performed . The setup allows learning from different sources.Allowing flexible learning process which doesn't compel an entity (ship) to be in service to learn. Using this network a Ship can learn from the behaviour of other ships as well.

Although, this is a simple representation of implementation in real life it gives us an idea about a skeleton around which a sophisticated system can be developed as a future work.

Since this is not the primary focus of out project we shall not go into the details of this , and move towards the experiments which shall help us to get a proof that the concept hereby mentioned actually works or not.

Chapter 5

Experiment Design and Tests

This chapter explains the details of the experiments performed in this project.Firstly the design of the experiment is presented.The chapter starts by exploring the details of how the DRL model has been made and what has been done to optimise the policy taken by the agent. Next the result from implementing a double DQN to the problem has been presented.

Several DRL models can fit the problem of policy optimisation, we need to explore them in the order to decide which one is the most suitable.These models are usually guided by fixed values , or control points which determine the efficacy of a model. These parameters are known as the hyperparameters. Although these can be optimised by testing all possible combinations with each other, this is a tiring and computationally intensive task. For example in a DL model there are parameters such as , number of layers in the network , nodes in each layer , activation functions , error functions , learning rate etc, just to name a few. Each of which have several options to choose from. Further more when a DL model is used in RL to constitute a DQN or similar network, there are other parameters which are added to this list of hyper parameters , these include epsilon , gamma , tau , replay batch size etc. In this project a more of a brute force approach towards hyper parameters optimisation ha been adopted. They were selected entirely on a hit and trail basis.

Here in this chapter we shall first discuss a case with power production vs Power storage ratio of 1:1 which means that the experiment was performed for a Engine with max power of N Kw along with an Energy storage of N KwH .To check if this case will work for another case also the battery capacity was then decreased making the Engine to storage ration 3:2. The Experiment was also designed in

such a way that the model understands only the dynamics of the working of the Engine and battery.

In Each experiment case was divided into two parts , based on the activation functions. Both the cases were tested for combinations of *tanh relu and linear* activation functions.The input parameters such as the load , battery level ,were noramlised between 0 and 1 using the minmax scaling , the reward was normalised between -1 and 1 and the output , i.e. the engine power level was a set of five states comprising $0\%, 25\%, 50\%, 75\% and 100\%$. The models were then analysed for the Exploration , training and Exploitation phases. Depending on the action taken the action space was divided into 'adjecttified' notations for easier understanding of the reader. finally results obtained from fuel consumption In each phase of various experiments was discussed.

5.1 Experimental Setup: DRL model

For the experiments to be performed we first need to discuss the architecture of the Experiments. This was previously introduced in a previous chapter , now we shall be discussing the details of these component.

These experiments shall be using the same DRL model, this 'base-model' has been modified to test the difference in the result obtained by changing the activation functions used in DL network.Components of this model are discussed below.

Each simulation consist of 1000 episodes and each episode has 90 time-step , each time step representing beginning of 1 minute interval in operation.

The simulation itself was divided into three phases , Exploration ,Training and Exploitation or Performance phase. Exploration phase consists of first 100 episode where the agent takes random decisions just to explore the environment , the episodes 100-600 consists of training phase . During this phase the agent linearly decreases the randomness in its actions and switches to using 'learned' behaviour from previous experience. Finally the agent uses this learned behaviour to see how it performs for the next 400 episodes in the Exploitation or performance phase. Keeping in mind that DRL is always learning , the minimum value of randomness in actions has not been made zero.

5.1.1 Load

Since this DRL uses DL to predict the value or Q function for the RL , the load has been normalised between 0 and 1.Other than this the load also need to represent all power levels of the action space , hence a generic load cycle was generated using the values mentioned in 4.1 which comprised of somewhat evenly distributed values. In real life these values shall lie between 0-100%- engine level. This load

was discritized for a time step of 60 seconds.Furthermore each load value was multiplied by a random value between 0 and 0.3 to obtain the next load cycle .This was done to introduce randomness in the load behaviour during a normal operation cycle. For each training episode this load cycle was repeated 3 times giving us a total load cycle of 90 minutes.

An Example of this load cycle has been presented below.

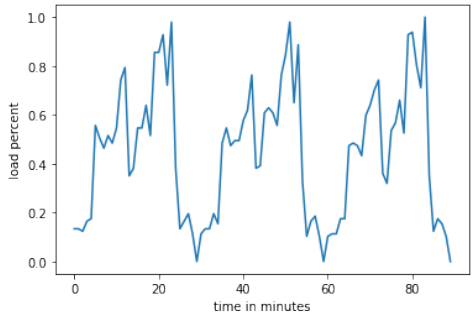


Figure 5.1: Normalised Load in each episode

This load cycle represents an operation distributed in 90 minutes ,each minute being a time step. This load cycle changes with each cycle and each episode.

This normalised load is one of the inputs to the model. the other input is the Energy storage SOC or state of charge.We shall now discuss how this was implemented in the Project.

5.1.2 SOC : State of Charge

This is the only other input to the model. This represents the State of charge of the Energy Storage, like the name suggests. This has too been normalised between 0 and 1 , where 0 represents complete discharge or empty and 1 represents totally full capacity.

The agent needs to deal with handling various SOC values. The aim was to see how the agent would react if the state space was randomly changed. Even though the load was somewhat cyclic , the randomness introduced in SOC values in the beginning of each episode may allow the agent to adapt towards a good amount of possible state values. In this way the agent should be able to learn how to act in several different circumstances. To facilitate randomness in SOC, Each training episode starts with a Random SOC value between 0 and 1. This has also been done to simulate cases such as Shore charging where the SOC suddenly jumps up from a lower level to higher level. Any other cases such as Auto discharge can also be

taught to the model in the same way.

During the Exploration phase of the Experiments the SOC inputs at the beginning of the episode was first done as a random choice between any values between 0 -30 and 80 -100% , for the rest of the episodes the SOC was chosen randomly. This was done exclusively to facilitate exploration in a way that the agent learns to interact with each reward function separately. This behaviour was also tested with complete random generation. The results were comparable with negligible difference in learning.

When the SOC and the load is known the model need to take an action, in the following subsection we shall discuss how the choice of actions have been implemented in the experiments.

5.1.3 Actions

The action taken by the agent is dependent on the above mentioned inputs and the Q value of each action corresponding to the action - input or state pair.The value or 'quality' of an action in this state pair is calculated using a Value function. In a DQN this the value function is represented by a DL netwrok.

Outputs

The actions are taken by choosing the best possible option corresponding to an action- state pair.While this value is calculated using the DL netwrok , the output themselves are very simple and represent a uniformly distributed discritized Engine state space. The agents job is to make a decision about what load capacity should it run the Engine at the beginning of each time step. These levels as mentioned earlier are 0 , 25 , 50 , 75 and 100% , each of then have a different SFOC. The SFOC as, mentioned in previous chapter gives a different value of fuel oil consumption for each action. The agents job is to chose the best action for each state pair. This value is determined using the function GET-ACTION() in the DQN network.

Epsilon greedy Algorithm for choosing action

As mentioned in the previous chapter , the agent learns from its own experience in a DRL. To make sure that the environment is sufficiently explored in the beginning of the training and later the experience from these explorations are effectively used without needing to re explore everything, we use the epsilon - greedy algorithm.

In this project the agent is allowed to discover the environment for first 100 episodes , here the value of epsilon is taken to be 1 , representing 100% randomness. then the randomness is linearly decreased to 10% for the next 500 episodes. The

randomness is finally maintained constant at 10% for the rest of the 400 episodes.

When the agent decides not to take a random action it turns to the DL model(value function) for prediction of the Best action using the values generated for a certain state. The agent does the same for every time step.

Value based action :

For each state pair (SOC and Load) the DL model predicts a best possible action. This prediction is based on the reward the agent would obtain by taking the action in the current state as well as the cumulative total reward in the resulting state.

In this project , the ARGMAX(or index of max value) of the prediction (values) from the DL model is chosen by the agent.

To simplify understanding these actions have been given adjectives as following

Table 5.1: Actions

S.NO	Action Index	Action_name	Engine load %	SFOC	Action Name
1	0	eng0	0	0	Battery Mode
2	1	eng25	25	1.36	Worst SFOC
3	2	eng50	50	1.18	Ok SFOC
4	3	eng75	75	0.86	Best SFOC
5	4	eng100	100	0.91	Better SFOC

When the suitable action is taken the state space changes. The resultant SOC may be different from the previous SOC.

5.1.4 Resultant SOC

The resultant soc is calculated on the basis of the action chosen by the agent and the corresponding load. This is a simple simulation which takes in the power distribution between the energy storage and Engine. This corresponds to the equation mentioned in the previous section 4.2.2.

5.1.5 Reward

The action prediction itself is a feedback loop associated with a term called Reward , this reward represents the quality of each action in a state pair. In this project the reward has been divided as three signals.

Depending on the resultant SOC from the Action the reward can be either of the

following two.

If the Resultant SOC is between 0-30 % or 80¨-100% the reward is a normalised value between -1 and 1 , depending on the change of SOC from the previous level. For example if the resultant SOC if 16% from a previous value of 15% the reward would be the positive as increase in SOC at this range is desirable. The opposite applies to SOC levels in range 80-100%

Overall the reward function in these ranges can be given by the equation

$$reward = \frac{(soc_{new} - soc_{current})}{0.016}$$

here 0.016 is the maximum possible soc change.

if the reward if the resultant SOC is between 31 -80% is the following:

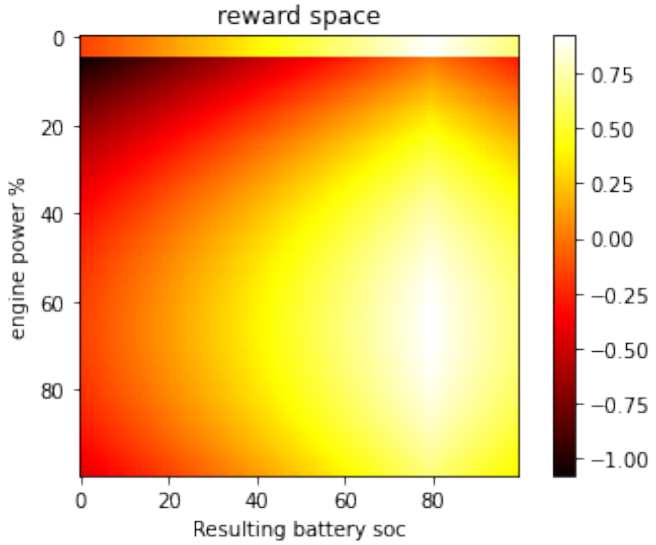


Figure 5.2: Reward Function

This reward function is a combination of the SFOC curve inverted and the resultant SOC shaped for a maximum value of 80%.

To get the corresponding reward for a state- action pair from the reward space shown above , the value of Engine power is compared from the Y axis and the resultant SOC is compared from the X axis.

5.1.6 Memory

The memory is limited space of 200000 , which stores the current States , actions , resultant state and the Corrsponding reward obtained.The following values are stored and used to train the model in the replay function :

1. Current SOC
2. Current load
3. Action
4. Resultant SOC
5. Next load
6. Reward

This memory is updated at each time step and cleared of older values as the new values are added.

5.1.7 Replay function

In the replay function a random batch of values are selected from the stored memory and Q learning algorithm is performed on these values giving us the values of each state. In this project the following replay function has been used:

```
def replay():
  batch_size=10
  for j in range(batch_size):
    if len(memory) < batch_size:
        return
    x=random.randint(0,len(memory))
    samples=memory[x-2]
    next_sample=memory[x-1]

    current_soc, current_load,action, reward, new_soc = samples
    next_soc,next_load,next_action,next_reward,next_new_soc=next_sample
    target = target_model.predict(np.array([current_load,current_soc])[np.newaxis,:])

    Q_future = np.max(target_model.predict(np.array([next_load,next_soc])[np.newaxis,:]))
    target[0][action] = reward + Q_future * gamma
    model.fit(np.array([current_load,current_soc])[np.newaxis,:], target, epochs=1, verbose=0)
```

in the replay function it can be seen that the a target value is calculated for the fist value of the chosen batch. This target value gives the Q value of all the actions for the current state. Q future or the Future state values is then calculated for the next sate from the target model. The value corresponding to the target and the action chosen is then made equal to the sum of current reward and a reduced commutative reward of next state.

A list is made corresponding to current load and modified target reward as labels

The model is then then trained on these lists as inputs. This is done for each values in the Batch selected.

5.1.8 Target train

To achieve convergence the weights from the model are then transferred to the target model after every 200th time step.

5.2 Test 1

The first Test was performed using the model stated above to a setup of 1:1 ratio of Engine power and Energy storage capacity . In this case these values are 75 KW and 75KwH.

The Experiment was divided into two parts , each part tested different combination of activation functions.Each part is then further distributed into Exploration , training and Exploitation Stage.

At first the results of all Episodes have been shown. Reward , their distribution , density have been presented. Same has been done SFOC and Actions.

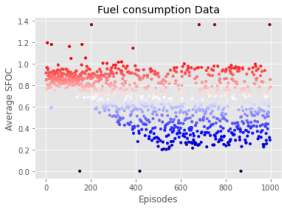
Later each stage result has been shown individually to further magnify the agent behaviour and actions, SFOC and rewards obtained at each level of simulation.

5.2.1 Test 1.1

This subsection presents the results obtained from simulating a 75 KW engine with a 75KWh Energy storage with a DL model with Tanh- Relu- linear activation function the DL model.

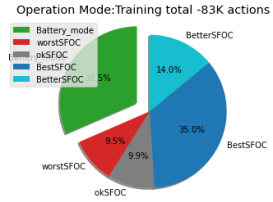
Overall training

The overall training represents all the episodes together. The following figures show how the model performs throughout the simulation process.



(a) SFOC for overall training

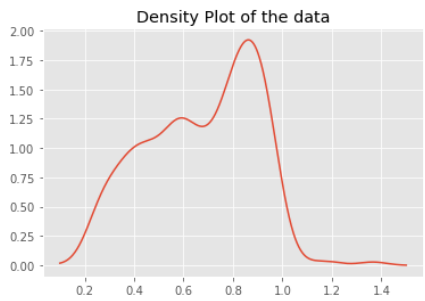
(b) Reward for Overall Training



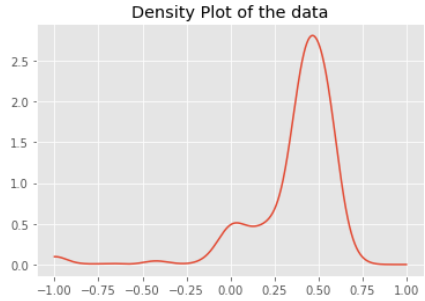
(c) Actions for overall training

Figure 5.3: Overall training results

The Density distribution of reward and SFOC throughout the process is given below :



(a) SFOC Density



(b) Reward Density

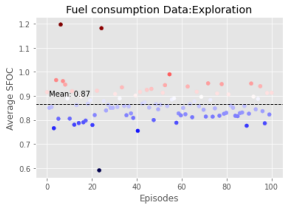
Figure 5.4: Density of results

This training has been distributed into three phases , Exploration ,Training and Exploitation.Recalling the simulation strategy we know that:

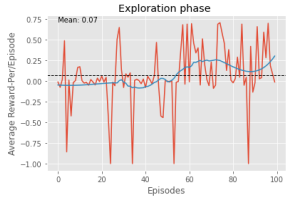
These stages have been distributed for 100 , 500 and 400 episodes. For the first stage the agent takes random actions , for the 2nd stage the values of randomness in action decreases slowly to 10% .In the final stage the agent uses knowledge from the previous two stages and exploits it to get the best possible action.At this stage the value of randomness is fixed at 10% ,meaning that 10% of the time the agent takes random actions. The next few subsections show how the model learns to map the behaviour of the system and use the experience to successfully teach itself how to optimise the policy or sequence of action to maximize reward and reduce SFOC.

Exploration phase

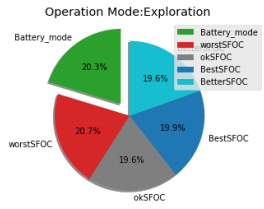
The first 100 episodes of the experiment consist of the Exploration phase. During this phase the agent takes random actions to see what kind of reward is obtained at various states for different actions.The following results were obtained in terms of average reward , SFOC and total Actions for Exploration phase.



(a) SFOC for Exploration



(b) Reward for Exploration



(c) Actions: Exploration

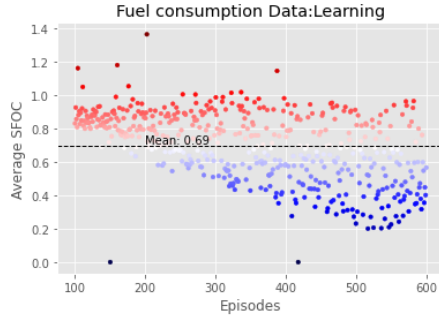
Figure 5.5: Exploration Phase results

Training Phase

As mentioned earlier In the training phase the value of randomness in choice of the action is slowly decreased. During this phase the agent learns to use the experience gained previously while slowly decreasing the importance of exploration. This Phase lasts for 500 episode and continues from the 100th episode.The rate of epsilon decay is calculated as 0.002 per episode.

We shall take also see how the action are improved over the course of training separately.

(a) Reward during Training

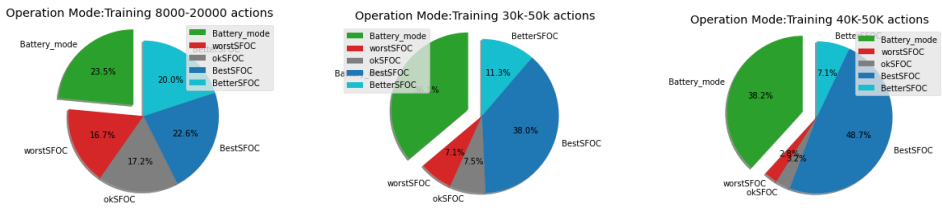


(b) SFOC during Training

Figure 5.6: Training Phase results

The actions taken during this phase clearly explains the behaviour of the agent in terms of learning , as the agent learns to get more and more reward as the training progresses. To visualise this clearly these actions have been distributed in smaller groups to study how the decisions are taken through out the training process. The total number of actions taken during each training is around 83000. First 8000 are explorations and the last 36000 are Exploitation actions. Thus we shall now present the result excluding these two phases as they have been presented separately.

These results have been presented in a sequential way in appendix. however it has been shown here the distribution of actions in the beginning , the middle and the end of the training phase:



(a) Beginning of training **(b)** middle of Training **(c)** End of training

Figure 5.7: Actions during Training

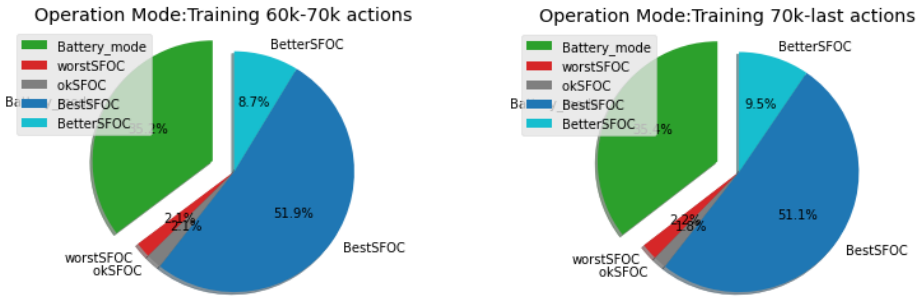
Finally the trained model was tested for remaining 400 episodes

Exploitation phase : Performance

During this phase the model uses learned behaviour from the previous two phases. During this phase the probability of taking random actions is only 10%

We can assume this phase to be the result of the Training. The following shows how the model would fare when put up with the normal load profile. The results of this phase can be taken as the output of this model. Even though the Model is primarily sticking to the policy leaned during the training there is still a 10 % chance of the model taking random action. This gives the model a chance to learn new behaviour , but also a chance to deflect from behaviour which might possibly be almost ideal in a given load scenario. The results of model performing in this phase is given below.

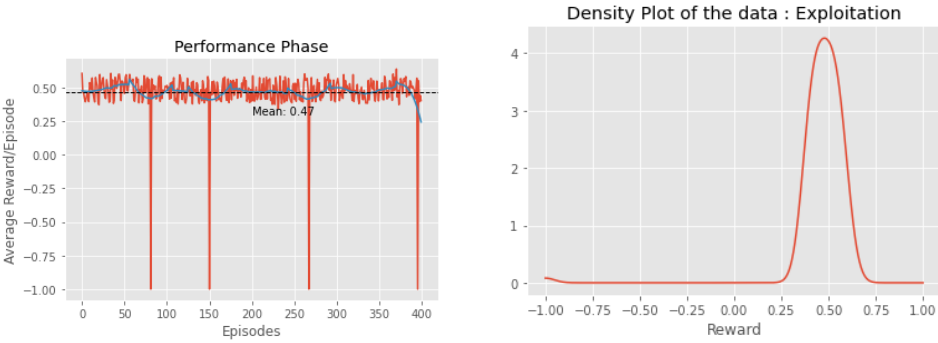
First to visualise how the model performed during this phase , we shall visualise the Actions taken during this phase. Models Ideal behaviour would be to have a good combination of 'BEST SFOC' and 'Battery mode' actions while trying to minimise Bad and Ok SFOC actions . The following figure shows the details of actions taken during the 60-70k , and 70k - last action.



(a) **(b)**

Figure 5.8: Exploitation Phase actions

It is also necessary to visualise the Reward during this phase. The reason behind this is that , even if the actions taken during this phase are good , the consequences of these actions may not be good. In other words the the agent may drive the system out of range, by over or emptying the battery.



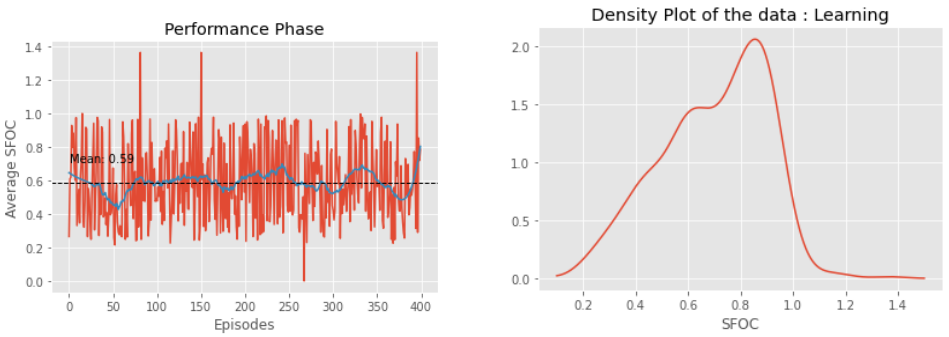
(a) Exploitation phase Rewards **(b)** Exploration Phase Reward Density

Figure 5.9: Exploitation Phase Reward Density

As mentioned and can be seen in figure above , that there were four instances where the model chose to take actions that led to over of under charging of the battery.

The Visualisation of SFOC during this period is important as It allows us evaluate our model on the basis of Economic and Environmental grounds. Maintaining Lower SFOC is the desireable behaviour of the Engine , however It might only be a good representation of ideal model behaviour in certain conditions. For example if the battery SOC is very low the IDeal behaviour of the model should be to

Increase the SFOC as soon as possible , which means running the engine at max capacity. However this leads to SFOC higher than the Ideal 75% capacity.



(a) **(b)**

Figure 5.10: Exploitation Phase actions

With visualising the SFOC, Rewards and the Actions together we shall be able to judge the model. While Actions give us an overall Idea of how the model fairs , by telling us which actions were taken and if the models initial behaviour is satisfactory. The Rewards during this Phase are able convey the fundamental nature of the model behaviour , that is to maintain good engine operation while maintaining a lucrative range of Energy storage capacity. Finally the SFOC visualisation is able to tell us more details regarding the Fuel consumption and overall efficiency of the model in maximising the utilisation of the fuel and reducing emission as a result of extracting maximum out of the fuel as usable energy.

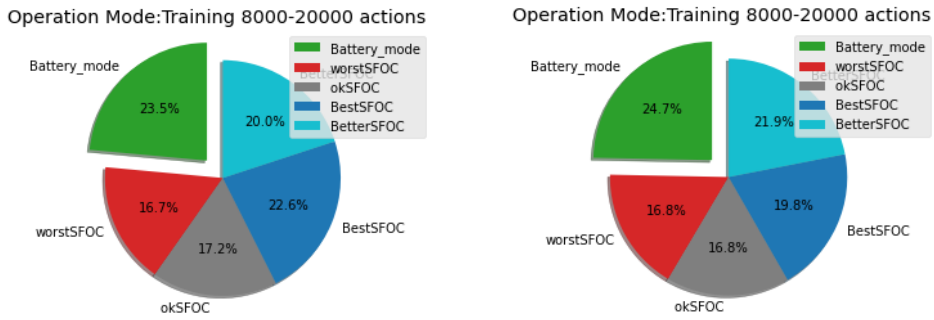
5.2.2 Test 1.2

In this section we shall show a comparison between the previous model which used Tanh- Relu-activation functions , and this model which differs only by the fact that it uses a Tanh-Tanh activation function. While the difference does not seem so significant in terms of structure, the agent behaviour is considerably different. We do so to understand more about the working of the model , but also to enable us to make a choice when it comes to desired actions we want to embed in our final model. We shall first start by comparing the models behaviour during the training Phase in terms of actions , SFOC and Reward and then finally make a comparison between the actions taken during the exploitation phase.The description of the results and their comparison has been done later in the report , in the discussion section. A detailed description of the results can be found in the appendix.

Training phase

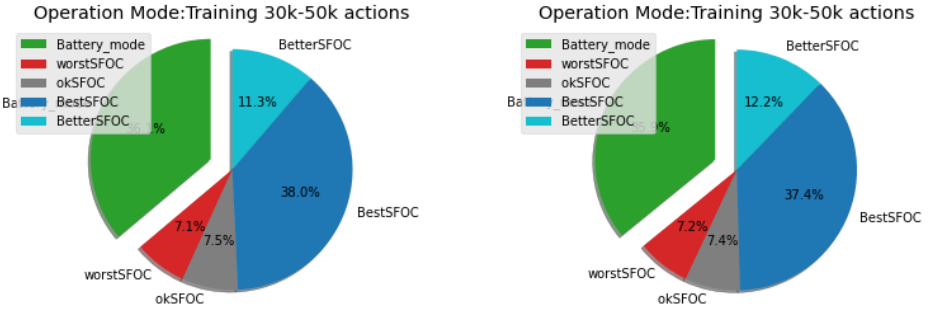
Since the training parameters during the two experiments remain relatively the same , it is only logical to see the training of the models(with different activation functions) and compare them. This will give us an idea about efficiency of the training and allow us to visualise the details of the relevant judgement factors.

At first we shall see a comparison between the actions taken during the different stages of training.



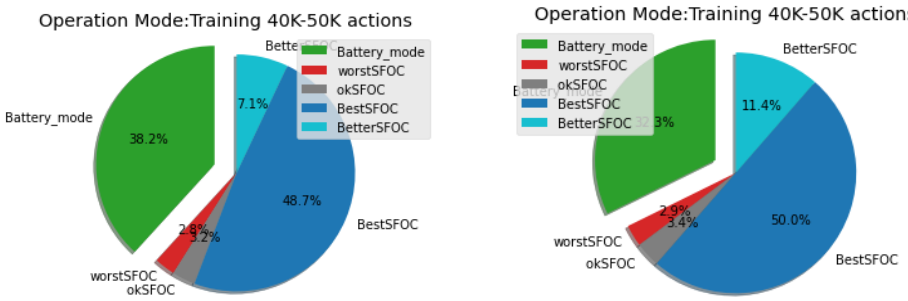
(a) **(b)**

Figure 5.11: Comparison between initial training



(a) **(b)**

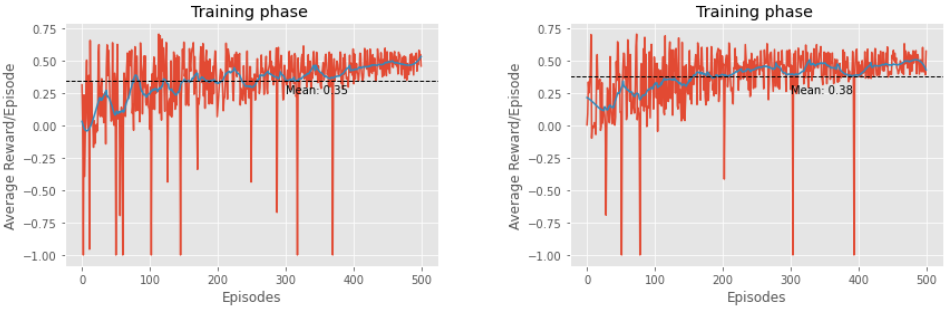
Figure 5.12: Comparison middle training



(a) **(b)**

Figure 5.13: Comparison End of training

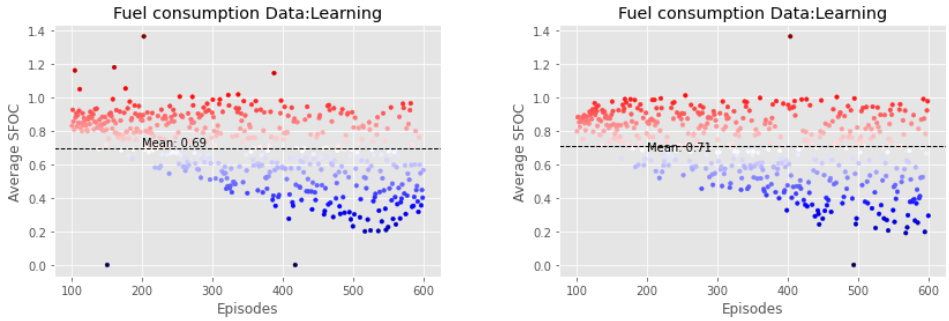
The figures above give us an overview of the training results , however to properly visualise the behaviour of models in terms of Storage capacity behaviour we also need to compare the rewards we have obtained during the training process.



(a) **(b)**

Figure 5.14: Comparison of Reward during training

Finally we need to visualise the training of models from a fuel economy prospective by comparing the SFOC.



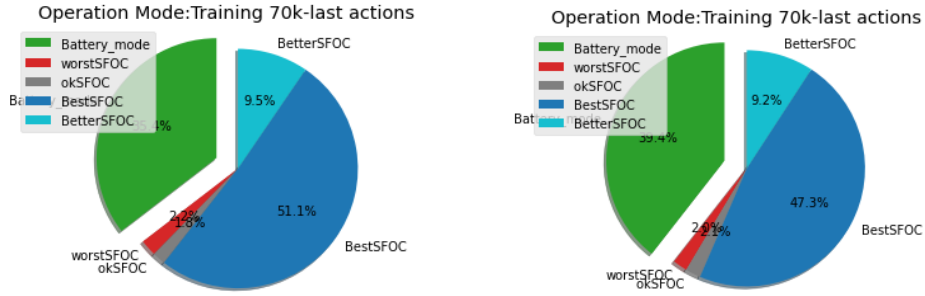
(a) **(b)**

Figure 5.15: Comparison of SFOC during training

Exploitation phase : Performance

The comparison between the results obtained during the Exploitation phase has been done in this subsection. Here we will be able to see how these two models compare to each other in terms of nature of actions taken,Energy Storage utilisation and Fuel economy.

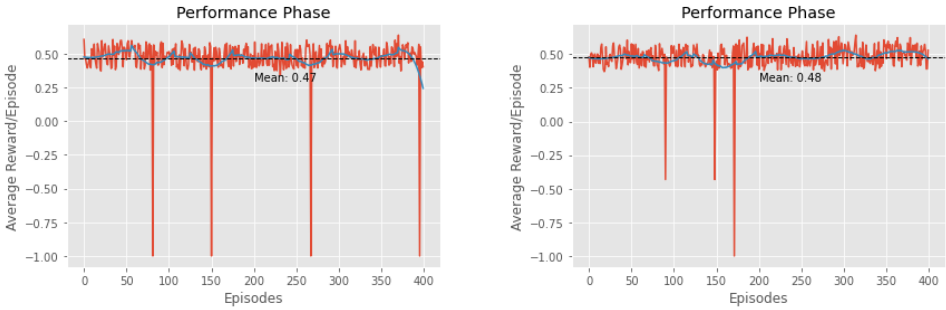
First we will see the comparison between action taken by the agent in the two cases. This will give us a brief idea how the agents compare to each other as an overview.



(a) **(b)**

Figure 5.16: Comparison of actions during

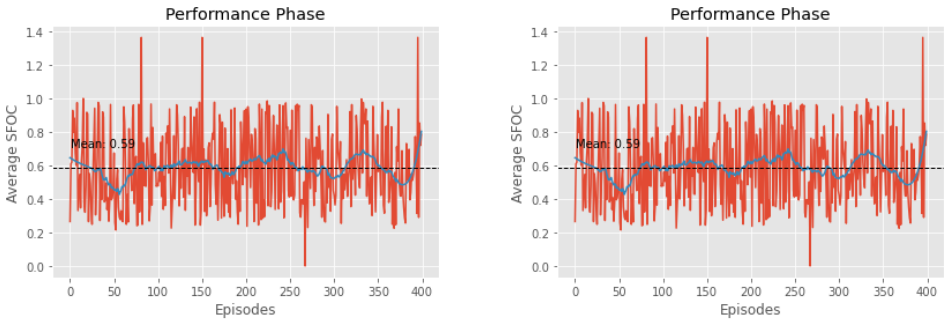
To visualise storage capacity utilisation we shall compare the rewards the agents obtained during the exploitation phase.



(a) **(b)**

Figure 5.17: Comparison of reward during Exploitation

Finally to compare the fuel economy of the agents we shall now compare the results between the SFOC during this phase.



(a) **(b)**

Figure 5.18: Comparison of SFOC during Exploitation

5.3 Test 2: Reduced Energy Storage Space

To be able to safely check the versatility of the model , this experiment will investigate the working of the model under slightly different system. In this experiment the capacity of the Energy storage will be reduced so that the ration between the Engine and the Energy storage is 4:3 This this will help us see how the model performs when there is a drastic change in the dynamics of the system.

Firstly the effect of each action on the soc will increase , and consequently the reward gradient between two time intervals should increase as well. We shall do a comparison between the 1:1 model , in both tanh and Relu -tanh cases individually and also individual results of this experiment.

5.3.1 Training comparison

Results below show the comparison between all four models tested. We begin by comparing the three phases of simulations , each of which has been compared in terms of action , reward and SFOC characteristic during that phase.

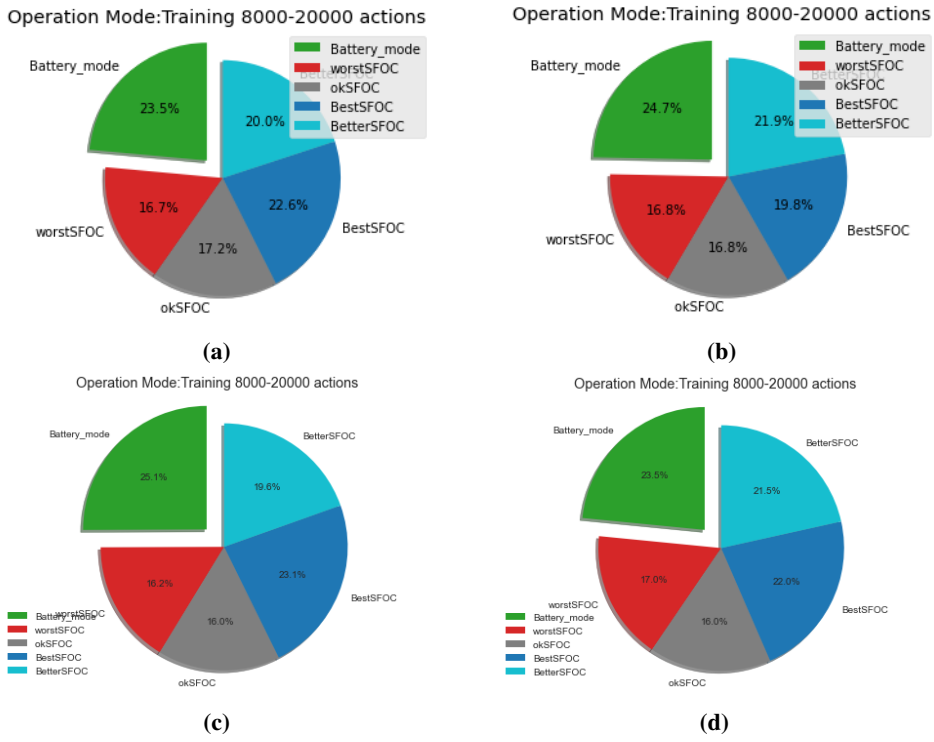


Figure 5.19: Comparison of Actions in the beginning of Training

[1]In the beginning of the training all the models are taking random actions but it is curios to learn how this training progresses in different cases and which model achieves stability faster.

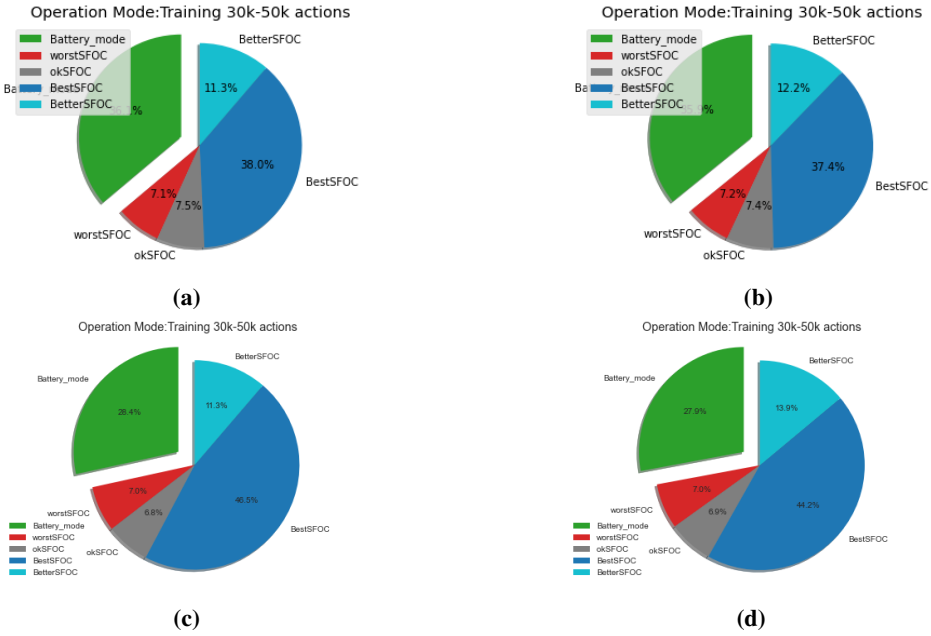


Figure 5.20: Comparison of Actions in the middle of Training

[1]To avoid repitative clutter in captions of figures the reader is advised to view the results in the following way - all parts names (a) represent result from tanh-relu-linear DL model in 1:1 Engine ,storage setup ; all parts named (b) represent tanh-tanh-linear DL model in 1:1 Engine , Storage setup; all parts names (c) represent result from tanh-relu-linear DL model in 3:2 Engine , storage setup and all parts named (d) represent tanh-tanh-linear DL model in 3:2 Engine

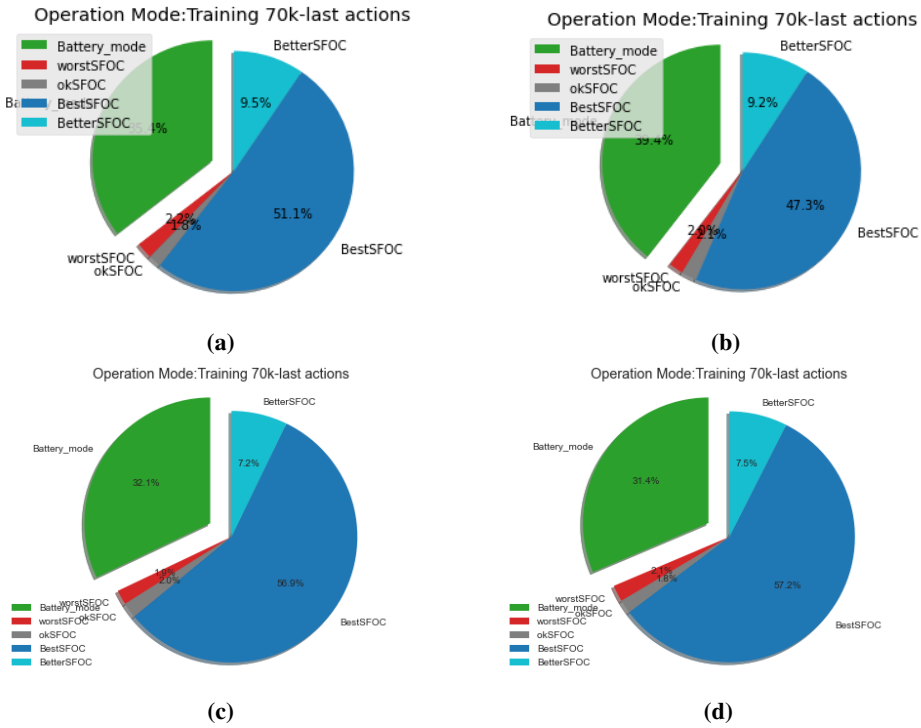


Figure 5.21: Comparison of Actions by the end of training

We must also take a look at the rewards during the training period , this shall give us a detailed view of how the models tend to learn to maintain the optimum energy storage levels.

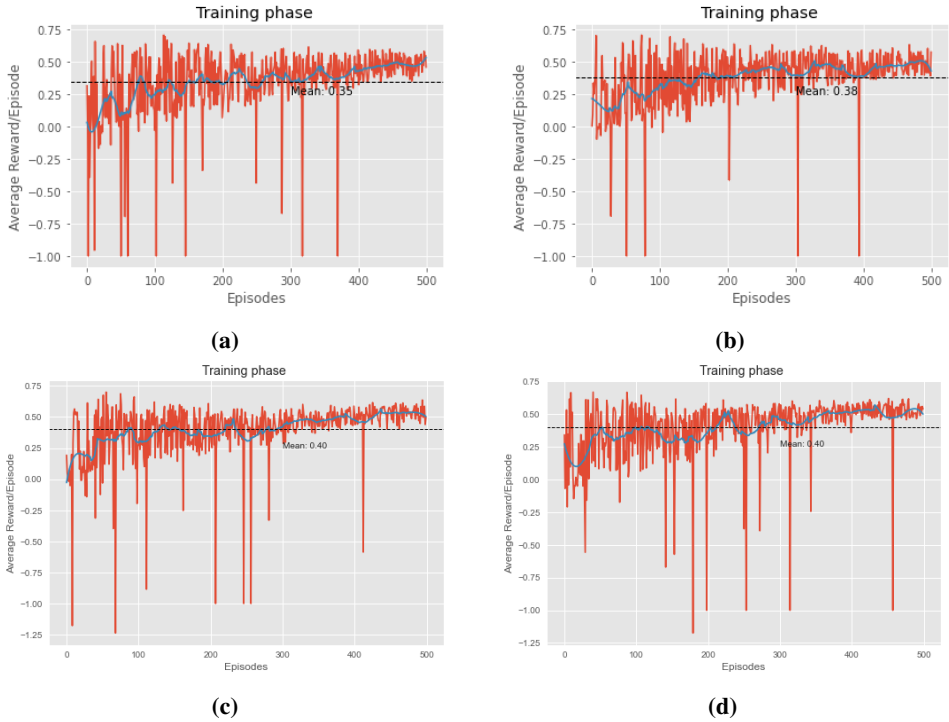


Figure 5.22: Comparison rewards of training

finally the SFOC of all four cases during training has been compared below.

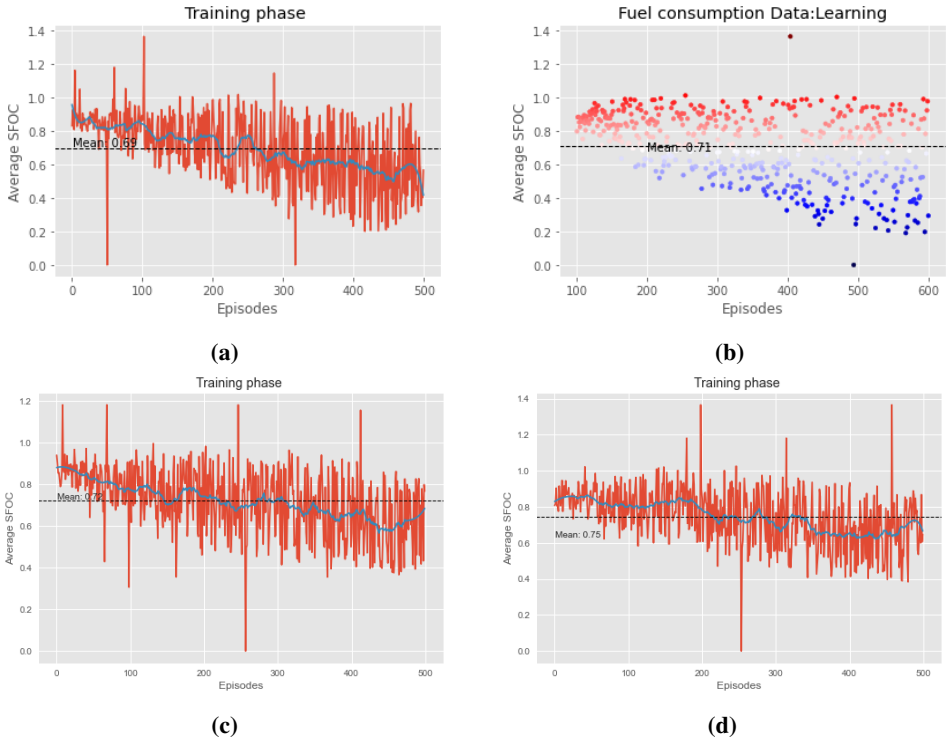


Figure 5.23: Comparison SFOC of training

5.3.2 Performance Comparison

To compare the performance of all the four models , it is ideal to have a side by side result visualisation of all for results. In this subsection we shall see the comparison between the actions , rewards and SFOC of the models in the performance phase. The parameters of the simulations remain the same hence the results are comparable . This will conclude the experiment part of this project and we will then move forward to discussing the results in detail and engraving the conclusions. Below are the comparison of actions taken by all four models during the Performance phase of the experiment.

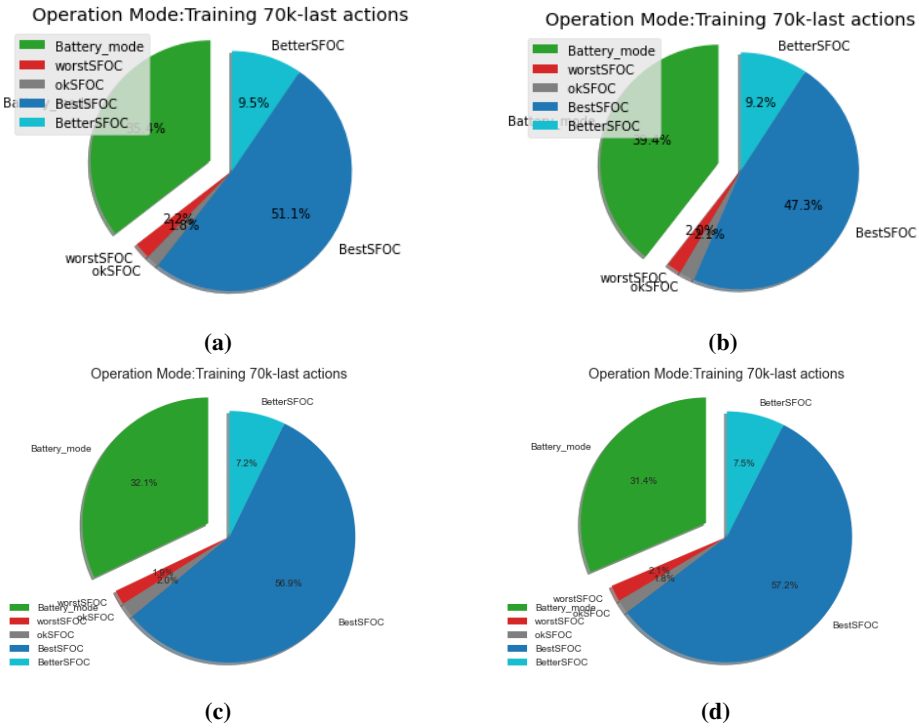


Figure 5.24: Comparison actions during performance

Furthermore the Rewards during this phase of simulations are presented below:

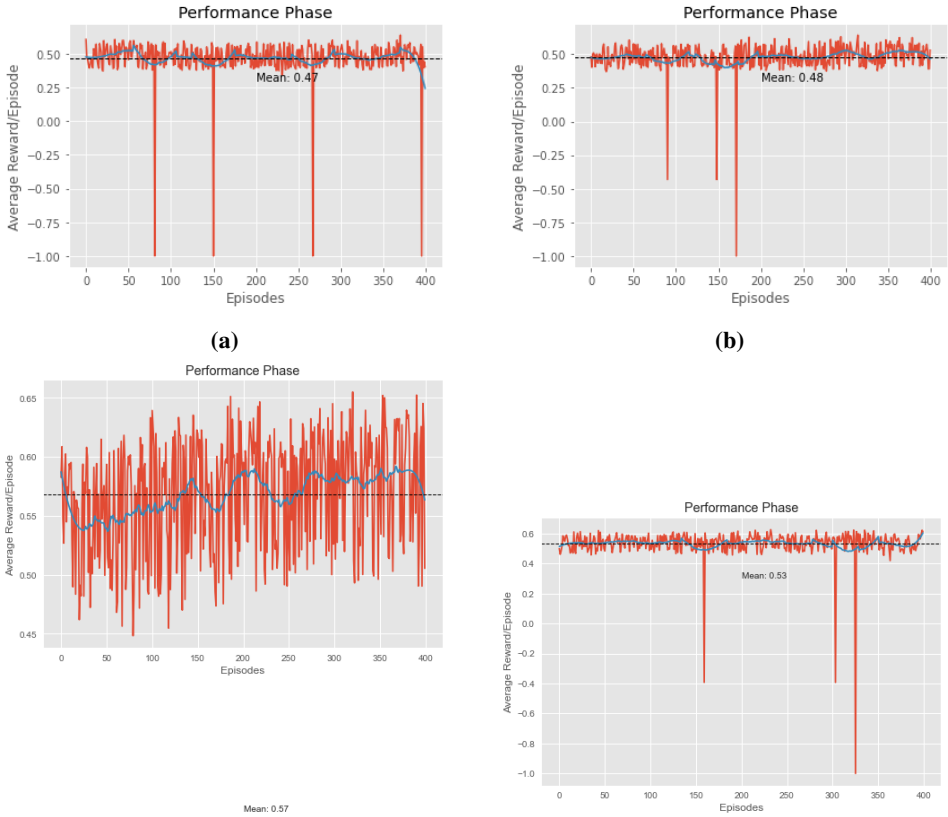


Figure 5.25: Comparison Reward during performance

It is interesting to note that experiment from 2.1 has never allowed the reward to drop below 0.4.

Finally the SOC from the experiments shall be compared giving us the idea how the Models faired in terms of fuel economy and Emission

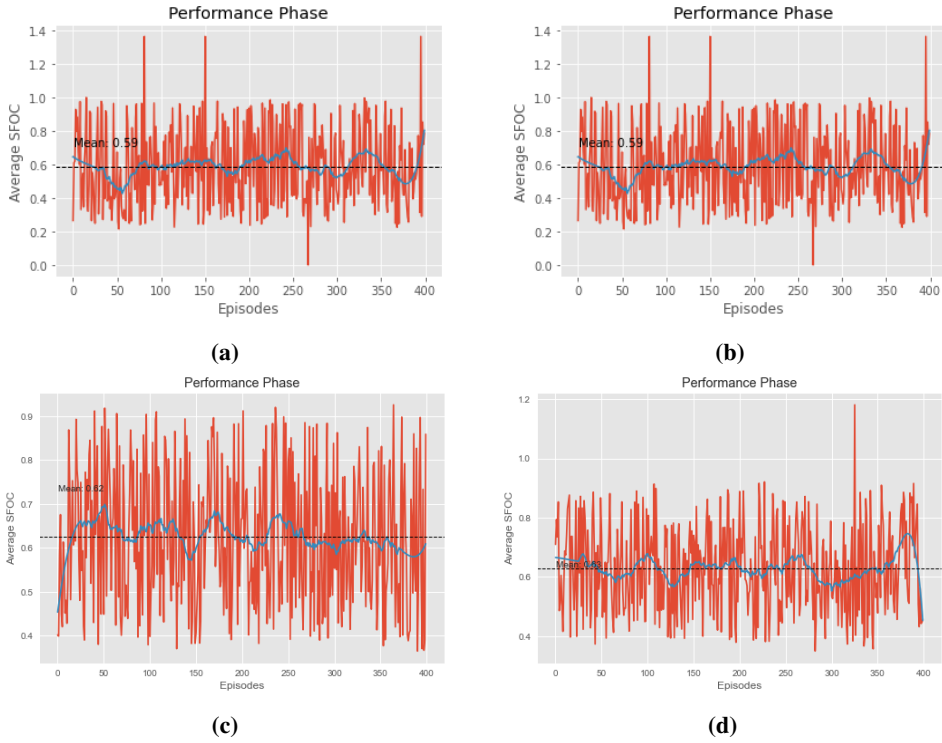


Figure 5.26: SFOC comparison during Performance phase

Chapter 6

Discussion

In this chapter the results from the experiments shall be discussed.

6.1 Experimental Setup

The process of Experiment design showed that it is possible to design an Operation controller, for an Energy management system in a Hybrid powered Ship. What has been generally used for employing control for virtual environments such as video games was used to employ active, online control in real life. The system and related states were studied and transformed into usable MDP parameters. A common DRL model was constructed whose parameters were tweaked, to study the difference in control behaviour.

6.1.1 States

It was found out that even though , most engineering systems employ continuous state variable, for employing control , it is possible to discritize the state space in the order to obtain reasonable control. It was also found out that the parameters such as the Load, and other states involved in the MDP have to be normalised between value of 0 to 1 in the order to support the Deep learning. It was also found out that to induce intelligent behaviour , a certain degree of cyclicity and randomness needs to be present in system behaviour.To achieve this A base load profile was created and repeated for three times.Each of this repetition represented a 30 minute cycle , making an overall cycle of 90 minutes of data. This was designed to simulate a round and half cycle of trip between two stations, but also enough data points in the cycle for a total exhaustion of energy storage space in an episode. Moreover each load repetition in the operation profile was given a certain degree of randomness, this was done to introduce stochastic nature in the

load profile. The load profile itself was not changed because in a real life operation ,usually routes taken by the ferry remains same between two stops and tend to have a relatively similar paths and trip duration. However it was decided that a variation of randomly generated 30% variation shall be introduced in the load in the order to make the model learn the dynamics of the system. In this way the control model may take help from the load value, but not entirely depend on the load cycle, and also does not need a load prediction in the order to work , as it has been proven in previous literature that such prediction models tend to have varying degree of success in real life scenario. This can very well be related to how a human learns to walk , it has to figure out the dynamics of walking , such as balancing, speed and sense of direction and not rely on memorising or predicting behaviour based on the path it takes. More over as human learns intelligent behaviour it is able to perceive what it sees and makes judgement based on what it sees and the feedback it receives from performing an action.

Other than the load, an important parameter of the system was the SOC or the state of charge of the energy storage device.Along with the load, the DRL model was made aware of the the SOC which is dependent on both load (unknown variable) and action (known variable). Based on these two the model was asked to take suitable action at each time step. Just like the load, SOC data was also normalised between 0 and 1. This was done to allow the model to learn behaviour without any bias between the states. With these parameters being the only input the model is given, it was asked to take a decision.The model does so by choosing a series of action. Each action being the best possible, or near best action considering the feedback it received from previous actions and the current perception of the environment.

An another important note on the design of the SOC for the DRL model was that, the charging and discharging models not being considered in this experiments, the energy storage devices tend to have a time delay for charging and discharging. This time delay varies from device to device,and depends on several parameters, like type of materials used in storage, technology used , age etc. And such data is not only very hard to model, but scarcely available. Instead, the battery model was designed with zero lag and instead such phenomenon and discrepancies were assumed to be added in the load itself. For example if the age of the battery causes charging faults, it would require longer to charge, or higher power input. This extra load can be added to the load profile itself.

6.1.2 Actions

Based on the inputs the model gets, it is necessary to give a number choice of action the model must choose from this is represented by the action space. In an ideal scenario this shall be continuous .However such level of control can only be achieved with highly tailored reward functions which are able to give the model enough sparse reward so that the agent not only learns good from the bad, but also how good and how bad. Modelling of such reward functions suffer from several shortcomings such as personal bias of the designer and also pertain to the fact that these models seldom represent the actual phenomenon and are tediously hard and mathematically complex.

In this project it was assumed that the action space consists of five actions. They were equally spaced and distributed between 0 - 100% engine power. This was done to give the model enough chances to chose good as well as bad actions. Even though each load value had its range withing the power levels of the engine capacity, the model was expected to learn that, it does not need to fulfil all load needs only through engine ,and it can use engine and storage power in varying proportions.

Moreover each action was assigned an index in an orderly manner, even though this is not supposed to have any effect in the learning and the performance of the model, it made the visualisation and simulation of the model easier and analysing the results fast and agile.

It is also interesting to note that model was able to learn the dynamics that differ these actions from one another. The actions had varying degree of effect on the environment and yielded varying levels of optimal behaviour. while 0% power level allowed the load to be satisfied by only the battery, and other power-levels partially or fully satisfied the load at any time step.

The mechanism of choice between these actions can be understood using the DL model. This model is basically a function that represents the value of an action corresponding to a state. To do this a dual DQN model was created. The model itself is a simple Dense model with two hidden layer as described in previous chapter. This model takes the SOC and load values and gives the best possible action. The aim of the model was to maximise the reward which we shall be discussed in the next subsection.

6.1.3 Reward

Designing of the reward was undoubtedly the most challenging part of the experiment. Normally the reward in a DQN is sparse and only signifies success or

failure of a whole episode , but in this case most of the situations signify any action representing success , resulting in need for an additional reward function.

Firstly the reward needs to be able to mould optimal engine behaviour. This was signified using the SFOC values , which represent how well the engine is able to utilise the fuel at any power level.

Moreover the agent responsible to utilise the Energy storage space with most efficiency. Hence it was necessary to add the component of Energy storage into the reward signal as well. Further this reward signal should be able to keep the agent from choosing actions which prevents SOC from going out of bounds, hence also preventing over and undercharging. This was done by creating a 2D reward space which had Engine SFOC VS Power level on one axis and resulting Battery SOC on the other. The reason behind choosing Resultant SOC and not the current SOC is that, the agent should gain knowledge of what affects does its choice of action have on the environment as well, and not base its decision on current environment entirely i.e. Load and Current SOC. which seemed a more 'human' way of looking at things while making decisions.

Further in the experiment it was realised that there is need to separate the Reward for the more critical zones of SOC , so the agent knows when to prioritise optimisation of Energy storage optimisation. This led to introduction of another reward signal that simply allows the agent to optimise only the Energy space within certain limits of battery SOC. This was done by making the Reward signal proportional to just the resulting SOC within 0-30% and 80-100% . We shall see later that even though this reward signal represent 50% of the reward space , the model successfully remembers the optimal engine behaviour.

Another modification to the Reward signal was done for the subset of reward space in 0% Engine space region. It was seen in previous reward signal that simply adding the Engine reward based on SFOC and the SOC optimisation reward led to the a 2D space where there was no significant indication to where the agent knows when to take the 0% engine power action. Hence a small strip of reward was added to the reward space for the 0% engine power to prioritise this level at higher SOC levels.

As a result of this reward space the agent learned good behaviour from a rather dense reward function which more or less helped the agent to understand the dynamics of good vs bad behaviour in terms of the a hybrid energy power plant.

6.1.4 DQN model

In this project a Simple Dual DQN model was tested , other than the MDP parameters there was a necessity to model some supporter functions that allows a DQN to perform control of a model representing a physical process via simulation. These include the memory and replay function and the simulation of the process.

While it was necessary to store previously experienced data , it was also necessary to be able to use this data to continuously make the model learn. Moreover there was also a need to have information regarding the effect of the action on the environment.

The memory constituted of 200000 units of storage, each unit being able to store necessary values of current &future states, reward and action taken at a particular time step. Since the memory is limited and model is continuously learning, it is also necessary to flush out older memories to make space for the new learning. This was done using *Deque* from *collections* in python which is a library used for simulation of storage such as the one use here. The model was able to store learning and experience data in this memory .

Using this memory to learn a good behaviour is one of the main tasks in a DRL network. This is done through a Replay function. This replay function was used to train the model from previously learned data while considering the type of reward it had obtained in that experience. To simulate this a random batch of memory was chosen and the model was fit for each set of data in the chosen data set. This allowed the model to create its own supervised learning setup in lack of data from any other external source. This also allowed to model to be able to perform better than any supervised data set, whose maximum efficiency is as good as the human operator of the system. The use of memory also ensured that the model has advantage over PID like or Rule based controls, which do not take into account the learning from previous experience.

It should be interesting to witness how the models behaviour varies significantly different from a normal PID like , or Rule based behaviour.

To demonstrate the affect of the actions taken by the agent on perceivable environment , it was necessary to create a simulator for the process. This simulator was designed to transform the difference between Load and Engine power resulting from the action taken into used units of the Energy storage. Even though this simulator was a simple energy transformation , it was able to give theoretical values of change in SOC. Once again it was assumed that factors like change in SFOC during switching between two engine power levels , losses in electrical and mechanical transmission and others , would be and can be compensated as variation in

load itself.

Effect of Shore Charge and Auto discharge

To simulate effect of Shore charging and auto discharge of the energy storage each episode simulation was made to start with a random SOC. This allowed the model to simulate a behaviour where the SOC would randomly jump to levels closer to 100% or 0% at any given point. Since it is tedious and impractical to assume that the charging and discharging of the SOC shall take place a certain time.

For example we shall see several cases where at the end of one episode , the resulting SFOC was at a lower value and the next episode started with a SOC closer to 100%; somewhat representing the Shore charging. In another cases we shall also see similar case but with lower SOC closer to 0% after the previous episode ended at a significantly higher SOC level.

6.2 Tests

In this section we shall discuss in detail the results obtained from employing various DQN models in online Energy management of a hybrid power-plant used in this project.

We shall first look at the model from performance prospective during exploring t, Training and performance phase. We will see how the models compare to each other first for a 1:1 energy ratio ,and then compare to a modified model with reduced energy storage.

This was done to see how does the model fare in two different scenarios representing similar setting. We shall discuss the significant differences between these models and how the DQN parameters affect the results from each model.

6.2.1 Exploration Phase

The simulation of model has been done for 1000 episodes ,out of this which the first 100 episodes have been dedicated to exploration, this means that the actions taken by the agent during this phase is entirely random and model is allowed to explore the environment. There are 90 time steps in each episode giving the model 9000 time steps for exploring the environment.

These 100 episodes have been further divided in two phases,; first 50 episodes have been dedicated to exploration of critical SOC regions. During this phase the model was able to explore the environment where the SOC at the beginning of the episode starts from a value between 0-30% or 80-100% ,. During this phase the agent was introduced only to the reward corresponding to battery SOC.

for the episode 50-100, random values of SOC was introduced. During this phase, all 4 models fared quite similar as actions were entirely random and there was no control over actions.

The model simply learned how good or bad each action for various points in the state space. This allowed the model to build the initial value function using the DL model. As it can be seen in figure 5.6 a , all the actions taken during this phase has about similar probability. However a significant difference was observed between first half and the second half of this phase.

This was due to the fact that initial episodes were started with values closer to the critical SOC values. Later the SOC was introduced randomly. It was observed that there were more instances of failure , ie the SOC running out of bounds during the first 50 episodes. Later this number decreases.

Upon further observation , it was seen that the average reward during this phase was closer to 0 meaning that, when the model takes random actions, equal amount of good and bad actions are taken. This was able to give us a value which we can compare to further in the simulation process.

The distribution of average SFOC was also random during this phase, pertaining to the fact that the actions were random as well. The value of average SFOC , for 1 episode when equal amount of good and bad decision is take 0.8.

It should be noted that is value is quite close to the value when a non hybrid power plant is running at optimal level, which clearly indicates that even if random actions are taken in a hybrid system, it utilised fuel as efficiently as a tradition non hybrid power plant working at optimal efficiency at all times. This only further strengthens the claims of superior energy efficiency of a hybrid Energy system in comparison to a traditional Marine diesel power plant.

The interesting part of the observation takes place during the Training phase. Here the value of randomness is linearly decreased from 100% randomness to 10% randomness.

6.2.2 Training Phase

The simulation of episode number 100-600 was termed under the term Training phase , as the model actually learns to modify its behaviour , and learns from previous experience to select actions that give it higher rewards. The difference between each models learning are clearly visible in this phase.

We shall first discuss the actions taken by each model as training progresses, and then compare the rewards and average SFOC , to see how quickly each model is

able to learn behaviour and how this behaviour vary from each other.

The training progress is monitored in phases , the total number of actions during whole simulations was calculated at around 83000 , out of which first 10% were neglected as they were the random actions in the Exploration phase. The final 40% were also eliminated as they constituted the actions in the performance phase.

Actions

When the actions taken by each model was compared during the beginning of the training phase, it was clear that the model was still taking more or less random actions as the probability of taking random actions were relatively high (action 8000-20000).

When the training progressed the models can be seen prioritising better actions, This means that the models were seen to significantly increase the instances where it either used battery power only or ran the engine at optimal levels.

The model was seen to reduce the instances where it took actions corresponding to 50% and 25% Engine power levels , which are associated lower SFOC . This proves that the model learns from the reward function that these Power levels are sub par , as compared to running the Engine at 0% , which has no SFOC , or is entirely clean , but when it decides to run the engine it chooses 75% power level majorly which has the best SFOC out of all levels and 100% power level which has significantly better SFOC as compared to 25% or 50%.

Finally to compare the results of the training we visualised the actions taken by the model during final training (70-83K). During this phase the model takes very few random actions (around 10%). Rest of the actions are taken from the experience stored during previous episodes.

All the models were able to significantly reduce bad actions to a very low percentage - of around 2% each and 4% in total. To see if these were random actions or learned actions we need to further investigate the simulation model in each episode. This will we done in the next subsection.

Further it is also necessary to compare the final trained model between 1:1 power model and 3:2 power model.

It can be seen in figure 5.22 that agent takes relatively lower battery power actions in the lowered battery capacity models. The number of bad actions still however remained almost same as before. This clearly indicates that the model was able to figure out that the storage capacity was lowered and it can not use the battery as much as it would have if the storage capacity was higher. It decides to substitute

the lowered capacity by mostly increasing the use of engine at optimal levels.

This proves that the model would be able to fair quite well even when the battery capacity is lowered due to any cause. This will not affect the overall performance of the over all energy system. This adheres to the fact that the engine efficiency doesn't change.

It should be interesting to see how the reward function needs to be changed according to Engine and battery health to obtain similar overall engine efficiency for a certain duration of time.

Reward and SFOC

Even though the actions taken in the training phase gave a brief idea about how the model is able to take better actions during simulation , it only represents the Engine and Energy storage usage. It can be entirely possible that even if the actions are good , they may be taken at the wrong sequence and the SOC crosses the critical bounds. To visualise this we shall now see how the reward develops over the training period.

Figure 5-23 Shows how the reward changes as the training progresses. Any instance where the reward is less than zero signifies a bad sequence of actions taken during the episode.

All four models were seen to significantly increase the reward as the training progresses.

In the exploration phase when random actions are taken the average reward is seen to be around 0 in all four models with several instances where the SOC runs out of bounds and the reward becomes -1 . This signifies that when random actions are taken there is good chance that the storage capacity runs out of bound.

In the training phase however all four models significantly reduce the reward , pushing it from 0 to levels around 0.6

Although the reward plot does not significantly tell much about the training and learning rate of the models , it is clearly evident that the instances where the reward reaches -1 , decreases significantly.

A major difference between learning characteristic of the 1:1 and 3:2 model is that, the former the rewards fluctuate more from the average level getting higher reward on both ends of the average level.Whereas the 3:2 model tends to stick closer to the average reward level. This may be due the fact that the reward gradient between two consecutive time steps is higher in this model as the actions have higher affect on the SOC due to lower capacity. It is also seen that the average reward in case

3:2 model is slightly higher as compared to 1:1 model.

To visualise the affect of SFOC on the average SFOC of the whole episode was calculated. This should be able to give us an Idea of how the model manages to utilise the fuel. This is clearly related to the actions taken, hence it is only relevant to discuss SFOC from a fuel utilisation prospective. It is not an Ideal way to analyse the results obtained besides when the comparison is done with a traditional power plant is done.

For example in an episode where the agent needs to increase the SOC of the battery , it seems only logical to run the engine at the maximum capacity. This will certainly yield a higher SFOC , as compared to running the Engine at 75% capacity. But this is not necessarily ideal in this case , as the agent also keeps in mind the SOC of the storage ,and tries to increase SOC levels as soon as possible.

However as seen in figure 5.24 there is a significant decrease in the average SFOC of an episode in all four models as the training progresses. With average SFOC pushed closer to 0.7 in all four cases. The average SFOC in case of lowered energy capacity case is slightly higher that 1:1 case , which seems only reasonable as the Agent decides to use more Engine than Energy Storage.

6.2.3 Performance

In this subsection we shall see how the trained models fared after training. This phase in the simulation is given in the action range 70k- 83k actions. This phase can be seen as the performance of the model in closer to a real life situation.We will first see the action distribution, followed by reward distribution and finally we will see what range of SFOC is obtained when a fully trained model is simulated for loads with random variation of 30% from a fixed level. In this phase the probability of taking random actions is still 10%. This has been done to allow the model to be still be able to explore any missed action state values, even though the probability of doing so is small, this ensures that when the simulation time reaches infinity the model will learn optimal behaviour.

Actions

The actions in this phase is very similar to the final training stage , in the sense that the the models take very few 50% and 25% Engine power decisions. However a significant difference in behaviour of Tanh- Relu - linear and tanh -tanh - linear model.

In the later case the model shows to use significantly more battery power only actions.

The behaviour of the model choosing to use more engine in the 3:2 case still continues in this phase.

The overall sense on actions in performance can be visually perceived in figure 5.25

The percentage of taking worse (25% and 50%) Engine actions is still limited to around 2% each where as the model runs the Engine at max capacity at around 9.5% in 1:1 case and 7.5% in 3:2 case. This may be due the fact that the agent realises the affect of taking this action results in higher SOC change which may not be optimal in some conditions, like when the SOC is approaching the 80% level from the lower side.

We shall now see how the Reward and SFOC levels are distributed in the performance phase.

SFOC and Reward

To make sure that the actions taken during the performance phase do not push the SOC beyond the bounds, we need to visualise the reward obtained.

The average reward during this phase was around 0.47 for the 1:1 model and 0.53 for the 3:2 models. The tanh-relu model was also able to give 0 instances where it pushed the SOC out of bounds in case of 3:2 model.

In 3 out the 4 tested models we see that the reward reaches -1 , at-least 1 time. To investigate the reason behind the individual episode needs to be studied. However this number is very low and considering total number of actions taken during this phase the percentage of taking such action is negligible. After deeper inspection it was found out that in all these cases , where the model allowed the SOC to go beyond bounds started with near boundary conditions i.e near 100% or near 0%. Further investigation showed us that this behaviour was either due to the model choosing random action or actions just before (or penultimate) pushing the SOC out of bounds and hence earning -1 reward. This is entirely possible but highly unlikely as SOC is fed into the simulation model randomly and the probability of taking a random action is only 10% in this phase. This tells us that this only happened because the model decided to 'explore' at the wrong time. Again the number of such instances is very low.

The average SFOC per episode for all four models were calculated around 0.6 , with lowest SFOC in 1:1 cases of 0.59. To give a prospective for comparison the best SFOC a tradition non hybrid diesel power plant shall give an SFOC of 0.85 even if it is run at optimal condition at all times.

To see how the model manipulates SOC from the beginning to the end of an episode , several cases from the simulation have been added in the Appendix1 and 2 . It is interesting to see how the model sometimes decides to keep on lowering the SOC of the storage even when the level comes down to an acceptable level and sometimes decides to maintain the level at an acceptable level.

Chapter 7

Conclusion

This research aims at exploring Deep reinforcement learning methods for the control of Energy management in Hybrid marine vessel. Based on the experiments performed it can be concluded that DQN can be used to effectively control behaviour of Energy management systems in order to increase Fuel usage efficiency and maximising Energy storage capacity. It can also be safely said that this research has potential to improve current Energy management Strategy.

Firstly the Model of a hybrid power plant was successfully transformed into a MDP model. This model consisted of a diesel engine with 75Kw capacity and an energy storage device. Policy optimisation was then performed using a DQN model. The model was distributed between a nomalised set of load and Storage SOC, equally spaced actions at varying engine Capacity, reward to maximise the desired behaviour and the agent in form of a dual DQN model.

The model was then simulated for a non consistent load profile of cyclic nature to simulate a 90 minute operation profile, at 1 minute time step. Two different models were tested for two cases of varying energy storage. These four models were tested and results were presented for each model. In each case the model was able to maximise desired behaviour. According to the formulated reward function, this desired behaviour was running engine at best efficiency,while maximising the usage of energy storage. It was found out that the all four models were able to achieve desired behaviour. In almost all the cases the model was able to opt for an action that either runs the engine at best efficiency or not run engine at all and use battery power. The model took either of the aforementioned about 85% of the time. Even the remaining actions were majorly consisted of running the engine at 100% power load , which indeed has better SFOC that running Engine at 25% or

50% which have significantly lower SFOC and considered as not as fuel Efficient. The percentage of these actions were lower than 4% in all four models. Finally the research questions are repeated and answered.

RQ1: How to Represent the problem of EMS Control for marine Hybrid power-plant as an MDP?

To represent the problem in question firstly all the components of the System were modelled in MDP terms. Environment consisting the state space was Identified. Reward functions was designed to induce desirable behaviour, a discritised action space was created and finally a Double DQN model was chosen for value function estimation and policy optimisation.

Finally these values were normalised to be effectively be used in a DRL method such as DQN which uses DL model to approximate Value function for state -action pairs.

RD2.Why and how DRL be used for policy optimisation to reduce Specific Fuel consumption during any operation cycle?

Dual DQN model was used identify behavioural dynamics of the System in terms of operation. This was model free approach that did not depend on any kind of external data.This gave a major advantage state of the DL techniques that requires large amount of data. DQN was identified as a better approach towards intelligent control also in terms of level of optimisation. While supervised learning methods can only achieve efficiency as high as the human level , even if the data is made available. On the other hand, DRL techniques such as DQN are independent of such restrictions and can achieve levels of optimisation much higher than human level.

DQN was employed to achieve high level optimisation during simulation. This consisted of a Replay function which was able to predict the value function for state actions pairs. This also helped us employ online learning in the control, which is continuously updated and learns from previous experience. Finally it was discovered that to achieve convergence a Dual- DL model needs to be used for value function approximation.

RQ3 Which Deep Learning model yields better compatibility with the given model of Reinforcement Learning?

All the models performed equally well with given system , however during this process a number of learning goals were achieved. At first it was discovered that DQN unlike other DL techniques consists of much more tweak-able parameters. It is computationally expensive and tediously time consuming to do a full grid ana-

lysis of these parameters. In this project 'take what works' approach was used to obtain a model that gave sufficiently good results since the DL model was relatively simple.

However the two different DL models were made for each of the two different system cases . These cases varied in terms of Storage Capacity. The first case consisted of a larger energy storage. The model was tested for both the cases. These models varied in terms of activation functions. Although no major difference in behaviour of these models was observed.

In each case the model was able to take good actions in terms of better SFOC and better utilisation of energy storage, performing significantly higher than a non hybrid power plant with similar capacity running at optimum conditions.

7.1 Thesis Contribution

Although DRL is widely researched area , most of its use is towards virtual cases such as video games and other virtual policy optimisation. To the best of author's knowledge this was the first attempt to employ DRL for Energy management in a Hybrid marine power plant making this thesis unique in sense of exploration of a new technique in this domain. Several concepts have been introduced in a general sense. The concepts used in the project can easily be adapted towards other cases as this attempt can work as a proof of concept for control problems involving physical systems.

Focused contribution of this thesis are:

- Introducing the concept of Deep Reinforcement learning based control for a hybrid marine vessel. This was an attempt to introduce intelligent control , which is able to optimise the operation of the vessel in terms of load allocation between the Engine and the Energy storage. This was done by simulating a DQN model to learn dynamic behaviour of the system.
- Design of a tailored reward function that was able to shape the DQN model in way that it is able to effectively choose good decisions irrespective of a load profile. This reward function was able to run engine at optimal level , as well as ensured that the Energy storage was utilised properly. This reward function was tested for two systems with varying Energy storage capacity, in both cases it was able to minimize actions which yielded lower efficiency in fuel utilisation i.e. higher SFOC.
- A git-hub repository which contains python notebooks and models that show implementations of the project in python.

- Suggested implementation for the implementation of the concept in real life scenario.

7.2 Future Work

The work in this thesis is presented as a proof of concept for using DRL in control of physical systems. This thesis was focused on energy management in a hybrid marine vessel. The aim was to suggest control techniques for energy management in a simple hybrid ferry , although the results seem promising and affiliate the potential to improve current techniques , such efforts need to done and be verified. Suggested below are some of the potential topics for future research related to this thesis and topic:

- Modifying the model to be used in a sophisticated simulator and consequently in a physical system seem like the next step towards development of the concept. The simulations in this thesis did not consider a particular charging or discharging model of energy storage. The efficiency of the engine while switching from one power level to other was also not considered. These factors are necessary considerations and can be done using a sophisticated simulator. This step is in necessary in terms of validation of the concept and check how the theoretical concept translates to real life scenario.
- Comparison between other state of art techniques such as fuzzy PID based control, load predictor control, SINDY models etc. and suggested method in the thesis for a case of Energy management and allocation in a hybrid power plant.
- Since there are so many systems that use hybrid energy , usually consisting of a power generator and an energy storage ,the possibility of comparison in terms of applicability is endless. One can compare various techniques and present the results.
- Training process of the model is far from being optimal, research has suggested use of concepts such as hindsight replay , dueling DQN, rainbow DQN to improve the training process of the model. Theses networks have shown to have advantage over the Duel DQN network used in this thesis.
- The state space considered in this discreet , which can be modified to a continuous state space using DDPG . Since most engineering models consider continuous action spaces, it should be interesting to see the comparison.

- Testing DRL for systems consisting multiple engines and multiple energy storage devices.Scaling the model for decision making in more complex scenarios such as multiple types of storage devices such as any combination of super-capacitors and batteries.This can also be modified for multiple ships, where each ship is considered a micro grid connected to each other through a floating or on shore power grid and each vessel can work not only as consumer but also a plant producing power for other.

Note: The access to simulation files / python notebooks can be achieved by Either sending Github collaboration request Github username in Subject to swapnilk0294@gmail.com or requesting a link for cloud based downloads.

References : Articles

[1] I. Abraham and T. D. Murphey. “Active learning of dynamics for data-driven control using koopman operators”. In: *IEEE Transactions on Robotics* 35.5 (2019), pp. 1071–1083.

[2] D. Ambuhl and L. Guzzella. “Predictive reference signal generator for hybrid electric vehicles”. In: *IEEE transactions on vehicular technology* 58.9 (2009), pp. 4730–4740.

[3] M. Ansarey et al. “Optimal energy management in a dual-storage fuel-cell hybrid vehicle using multi-dimensional dynamic programming”. In: *Journal of Power Sources* 250 (2014), pp. 359–371.

[5] F. Belletti et al. “Expert level control of ramp metering based on multi-task deep reinforcement learning”. In: *IEEE Transactions on Intelligent Transportation Systems* 19.4 (2017), pp. 1198–1207.

[6] D. P. Bertsekas. “A new value iteration method for the average cost dynamic programming problem”. In: *SIAM journal on control and optimization* 36.2 (1998), pp. 742–759.

[8] W. Böhmer et al. “Autonomous learning of state representations for control: An emerging field aims to autonomously learn state representations for reinforcement learning agents from their real-world sensor observations”. In: *KI-Künstliche Intelligenz* 29.4 (2015), pp. 353–362.

[9] H. Borhan et al. “MPC-based energy management of a power-split hybrid electric vehicle”. In: *IEEE Transactions on Control Systems Technology* 20.3 (2011), pp. 593–603.

[12] C. C. Chan. “The State of the Art of Electric, Hybrid, and Fuel Cell Vehicles”. In: *Proceedings of the IEEE* 95.4 (2007), pp. 704–718. DOI: 10.1109/JPROC.2007.892489.

[13] A. Charpentier, R. Elie and C. Remlinger. “Reinforcement learning in economics and finance”. In: *Computational Economics* (2021), pp. 1–38.

[14] A. Chasse, P. Pognant-Gros and A. Sciarretta. “Online implementation of an optimal supervisory control for a parallel hybrid powertrain”. In: *SAE International Journal of Engines* 2.1 (2009), pp. 1630–1638.

[15] Z. Chen et al. “Energy management for a power-split plug-in hybrid electric vehicle based on dynamic programming and neural networks”. In: *IEEE Transactions on Vehicular Technology* 63.4 (2013), pp. 1567–1580.

[17] N. Denis, M. R. Dubois and A. Desrochers. “Fuzzy-based blended control for the energy management of a parallel plug-in hybrid electric vehicle”. In: *IET Intelligent Transport Systems* 9.1 (2015), pp. 30–37.

[20] A. L. Ellefsen et al. “Online fault detection in autonomous ferries: Using fault-type independent spectral anomaly detection”. In: *IEEE Transactions on Instrumentation and Measurement* 69.10 (2020), pp. 8216–8225.

[21] R. Geertsma et al. “Design and control of hybrid power and propulsion systems for smart ships: A review of developments”. In: *Applied Energy* 194 (2017), pp. 30–54.

[22] K. Gökce and A. Ozdemir. “An instantaneous optimization strategy based on efficiency maps for internal combustion engine/battery hybrid vehicles”. In: *Energy conversion and management* 81 (2014), pp. 255–269.

[24] Y. Hu et al. “Energy management strategy for a hybrid electric vehicle based on deep reinforcement learning”. In: *Applied Sciences* 8.2 (2018), p. 187.

[25] J. Huotari et al. “Q-Learning Based Autonomous Control of the Auxiliary Power Network of a Ship”. In: *IEEE Access* 7 (2019), pp. 152879–152890.

[26] IMO. “ADOPTION OF THE INITIAL IMO STRATEGY ON REDUCTION OF GHG EMISSIONS FROM SHIPS AND EXISTING IMO ACTIVITY RELATED TO REDUCING GHG EMISSIONS IN THE SHIPPING SECTOR”. In: *UNFCCC Talanoa Dialogue* 72 (2018).

[27] IMO. “Marine Environment Protection Committee (MEPC), 73rd session, 22-26 October 2018”. In: *UNFCCC Talanoa Dialogue* 73 (2018).

[28] IMO. “Marine Environment Protection Committee (MEPC), 74rd session, 13-17 MAY 2019”. In: *UNFCCC Talanoa Dialogue* 74 (2018).

[30] T.-H. Joung et al. “The IMO initial strategy for reducing Greenhouse Gas (GHG) emissions, and its follow-up actions towards 2050”. In: *Journal of International Maritime Safety, Environmental Affairs, and Shipping* 4.1 (2020), pp. 1–7.

[31] E. Kaiser, J. N. Kutz and S. L. Brunton. “Sparse identification of nonlinear dynamics for model predictive control in the low-data limit”. In: *Proceedings of the Royal Society A* 474.2219 (2018), p. 20180335.

[33] K. Kavukcuoglu et al. “Human-level control through deep reinforcement learning”. In: *Nature* 518.7540 (2015), pp. 529–533.

[34] J. T. Kessels et al. “Online energy management for hybrid electric vehicles”. In: *IEEE Transactions on vehicular technology* 57.6 (2008), pp. 3428–3440.

[35] P. N. Kolm and G. Ritter. “Modern perspectives on reinforcement learning in finance”. In: *Modern Perspectives on Reinforcement Learning in Finance (September 6, 2019). The Journal of Machine Learning in Finance* 1.1 (2020).

[36] M. G. Lagoudakis and R. Parr. “Least-squares policy iteration”. In: *The Journal of Machine Learning Research* 4 (2003), pp. 1107–1149.

[39] C.-Y. Li and G.-P. Liu. “Optimal fuzzy power control and management of fuel cell/battery hybrid vehicles”. In: *Journal of power sources* 192.2 (2009), pp. 525–533.

[40] D. Li and S. K. Jayaweera. “Machine-learning aided optimal customer decisions for an interactive smart grid”. In: *IEEE Systems Journal* 9.4 (2014), pp. 1529–1540.

[41] T. Liu et al. “Reinforcement learning–based energy management strategy for a hybrid electric tracked vehicle”. In: *Energies* 8.7 (2015), pp. 7243–7260.

[42] C. M. Martinez et al. “Energy management in plug-in hybrid electric vehicles: Recent progress and a connected vehicles perspective”. In: *IEEE Transactions on Vehicular Technology* 66.6 (2016), pp. 4534–4549.

[43] M. R. Miyazaki, A. J. Sørensen and B. J. Vartdal. “Reduction of fuel consumption on hybrid marine power plants by strategic loading with energy storage devices”. In: *IEEE Power and Energy Technology Systems Journal* 3.4 (2016), pp. 207–217.

[44] C. Musardo et al. “A-ECMS: An adaptive algorithm for hybrid electric vehicle energy management”. In: *European Journal of Control* 11.4-5 (2005), pp. 509–524.

[45] A. Nair et al. “Massively parallel methods for deep reinforcement learning”. In: *arXiv preprint arXiv:1507.04296* (2015).

[46] F. Un-Noor et al. “A comprehensive study of key electric vehicle (EV) components, technologies, challenges, impacts, and future direction of development”. In: *Energies* 10.8 (2017), p. 1217.

[47] F. Odeim et al. “Power management optimization of fuel cell/battery hybrid vehicles with experimental validation”. In: *Journal of Power Sources* 252 (2014), pp. 333–343.

[49] G. Paganelli et al. “Simulation and assessment of power control strategies for a parallel hybrid car”. In: *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering* 214.7 (2000), pp. 705–717.

[50] S. Park et al. “Design and implementation of smart energy management system for reducing power consumption using ZigBee wireless communication module”. In: *Procedia Computer Science* 19 (2013), pp. 662–668.

[51] D. C. Parkes and S. Singh. “An MDP-based approach to online mechanism design”. In: (2004).

[56] D. Pudjianto, C. Ramsay and G. Strbac. “Virtual power plant and system integration of distributed energy resources”. In: *IET Renewable power generation* 1.1 (2007), pp. 10–16.

[57] M. L. Puterman. “Markov decision processes”. In: *Handbooks in operations research and management science* 2 (1990), pp. 331–434.

[58] X. Qi et al. “Data-driven reinforcement learning–based real-time energy management system for plug-in hybrid electric vehicles”. In: *Transportation Research Record* 2572.1 (2016), pp. 1–8.

[59] F. Qin et al. “Stochastic optimal control of parallel hybrid electric vehicles”. In: *Energies* 10.2 (2017), p. 214.

[60] P. Rodatz et al. “Optimal power management of an experimental fuel cell/supercapacitor-powered hybrid vehicle”. In: *Control engineering practice* 13.1 (2005), pp. 41–53.

[61] P. Rodatz et al. “Optimal power management of an experimental fuel cell/supercapacitor-powered hybrid vehicle”. In: *Control Engineering Practice* 13 (Jan. 2005), pp. 41–53. DOI: 10.1016/j.conengprac.2003.12.016.

[62] A. E. Sallab et al. “End-to-end deep reinforcement learning for lane keeping assist”. In: *arXiv preprint arXiv:1612.04340* (2016).

[63] S. Shah and S. Shahidehpour. “A heuristic approach to load shedding scheme”. In: *IEEE Transactions on Power Systems* 4.4 (1989), pp. 1421–1429.

[64] K. Shao et al. “A survey of deep reinforcement learning in video games”. In: *arXiv preprint arXiv:1912.10944* (2019).

[65] D. Silver et al. “Mastering the game of Go with deep neural networks and tree search”. In: *nature* 529.7587 (2016), pp. 484–489.

[66] T. W. Smith et al. “Third imo ghg study”. In: (2015).

[67] Q. Sun et al. “A multi-agent-based intelligent sensor and actuator network design for smart house and home automation”. In: *Journal of Sensor and Actuator Networks* 2.3 (2013), pp. 557–588.

[68] O. Sundström, D. Ambühl and L. Guzzella. “On implementation of dynamic programming for optimal control problems with final state constraints”. In: *Oil & Gas Science and Technology–Revue de l’Institut Français du Pétrole* 65.1 (2010), pp. 91–102.

[69] R. S. Sutton and A. G. Barto. “Reinforcement learning”. In: *Journal of Cognitive Neuroscience* 11.1 (1999), pp. 126–134.

[70] C. Szepesvári. “Algorithms for reinforcement learning”. In: *Synthesis lectures on artificial intelligence and machine learning* 4.1 (2010), pp. 1–103.

[71] V. Tatar and M. B. ÖZER. “The impacts of CO2 emissions from maritime transport on the environment and climate change”. In: *Uluslararası Çevresel Eğilimler Dergisi* 2.1 (2018), pp. 5–24.

[73] H. Tian et al. “Data-driven hierarchical control for online energy management of plug-in hybrid electric city bus”. In: *Energy* 142 (2018), pp. 55–67.

[74] C. Vagg et al. “Stochastic dynamic programming in the real-world control of hybrid electric vehicles”. In: *IEEE Transactions on Control Systems Technology* 24.3 (2015), pp. 853–866.

[75] C. J. Watkins and P. Dayan. “Q-learning”. In: *Machine learning* 8.3-4 (1992), pp. 279–292.

[77] M. Wiering and M. Van Otterlo. “Reinforcement learning”. In: *Adaptation, learning, and optimization* 12.3 (2012).

[78] S. G. Wirasingha and A. Emadi. “Classification and review of control strategies for plug-in hybrid electric vehicles”. In: *IEEE Transactions on vehicular technology* 60.1 (2010), pp. 111–122.

[79] S. Xie et al. “Pontryagin’s minimum principle based model predictive control of energy management for a plug-in hybrid electric bus”. In: *Applied energy* 236 (2019), pp. 893–905.

[80] D. Zhang, X. Han and C. Deng. “Review on the research and practice of deep learning and reinforcement learning in smart grids”. In: *CSEE Journal of Power and Energy Systems* 4.3 (2018), pp. 362–370.

[81] F. Zhao et al. “An overall ship propulsion model for fuel efficiency study”. In: *Energy Procedia* 75 (2015), pp. 813–818.

[82] F. Zhao et al. “Numerical study of soot particles from low temperature combustion of engine fueled with diesel fuel and unsaturation biodiesel fuels”. In: *Applied Energy* 211 (2018), pp. 187–193.

[83] F. Zhao et al. “Power management of vessel propulsion system for thrust efficiency and emissions mitigation”. In: *Applied Energy* 161 (2016), pp. 124–132.

[84] D. Zhou et al. “Online energy management strategy of fuel cell hybrid electric vehicles based on data fusion approach”. In: *Journal of power sources* 366 (2017), pp. 278–291.

[85] Z. Zhou et al. “A review of energy storage technologies for marine current energy systems”. In: *Renewable and Sustainable Energy Reviews* 18 (2013), pp. 390–400.

[86] Y. Zou et al. “Reinforcement learning-based real-time energy management for a hybrid tracked vehicle”. In: *Applied energy* 171 (2016), pp. 372–382.

References : Books

[7] D. P. Bertsekas et al. *Dynamic programming and optimal control*. Vol. 1. 2. Athena scientific Belmont, MA, 1995.

[10] N. Calder. *Marine Diesel engines: Maintenance, troubleshooting, and repair*. International Marine, 1992. Chap. 1, pp. 17–18.

[16] B. E. Conway. *Electrochemical supercapacitors: scientific fundamentals and technological applications*. Springer Science & Business Media, 2013. Chap. 2, pp. 31–32.

[18] O. Edenhofer. *Climate change 2014: mitigation of climate change*. Vol. 3. Cambridge University Press, 2015.

[37] M. Lapan. *Deep Reinforcement Learning Hands-On: Apply modern RL methods, with deep Q-networks, value iteration, policy gradients, TRPO, AlphaGo Zero and more*. Packt Publishing Ltd, 2018. Chap. 6, p. 135.

[38] M. Lapan. *Deep Reinforcement Learning Hands-On: Apply modern RL methods, with deep Q-networks, value iteration, policy gradients, TRPO, AlphaGo Zero and more*. Packt Publishing Ltd, 2018. Chap. 7, p. 163.

[52] M. R. Patel. *Shipboard propulsion, power electronics, and ocean energy*. Crc Press, 2012. Chap. 11.

[53] W. B. Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*. Vol. 703. John Wiley & Sons, 2007.

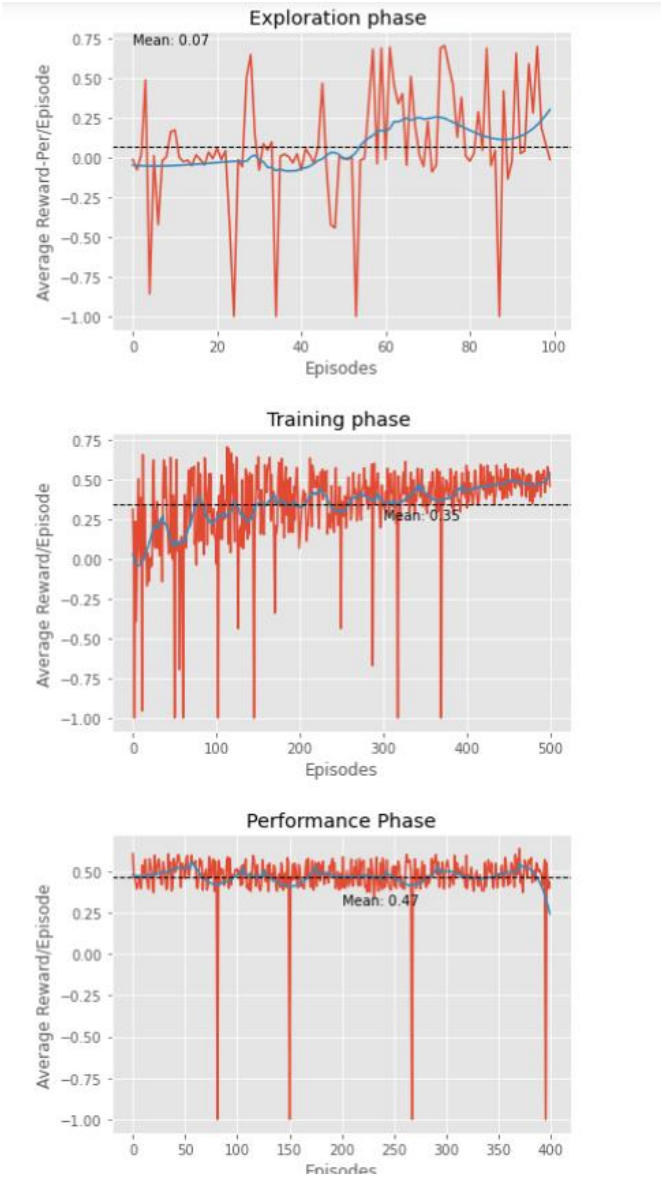
[54] W. B. Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*. Vol. 703. John Wiley & Sons, 2007. Chap. 5, p. 130.

[55] W. B. Powell. *Optimality Equations:Approximate Dynamic Programming: Solving the curses of dimensionality*. Vol. 703. John Wiley & Sons, 2007. Chap. 3, pp. 52–65.

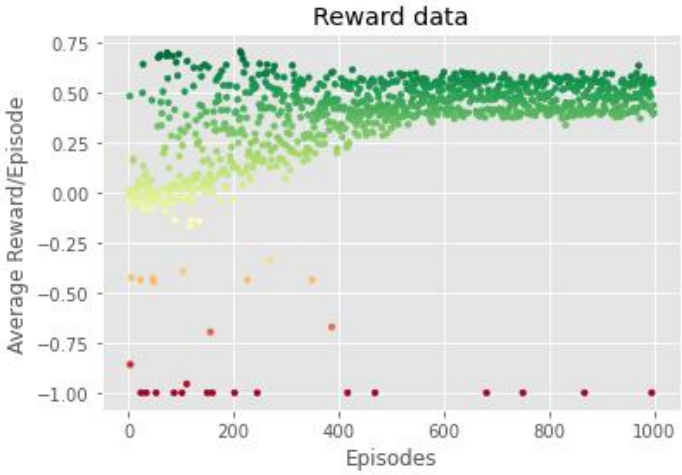
Appendix A

Appendix

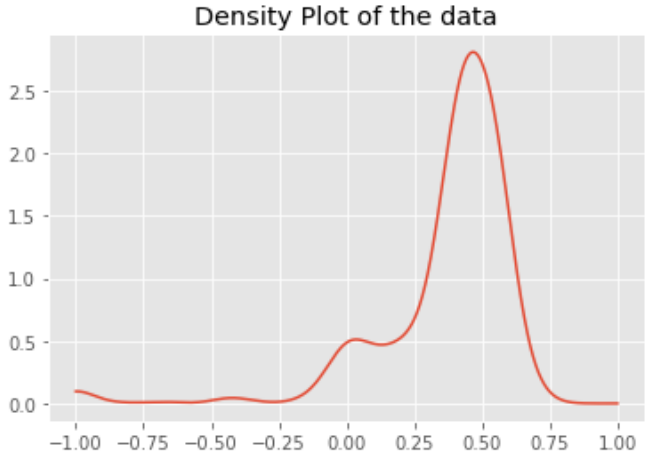
Experiment 1.1 : DRL with Tanh-Relu-Tanh



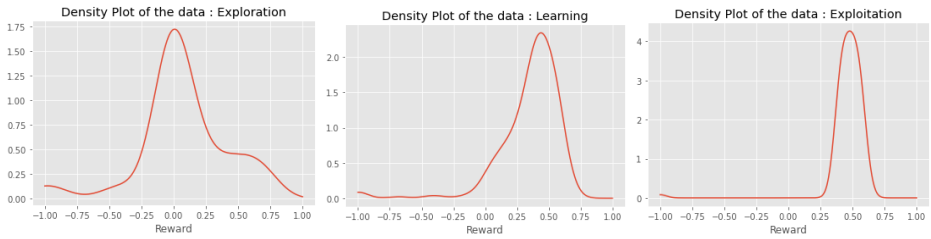
Figur 1: Rewards during different phases



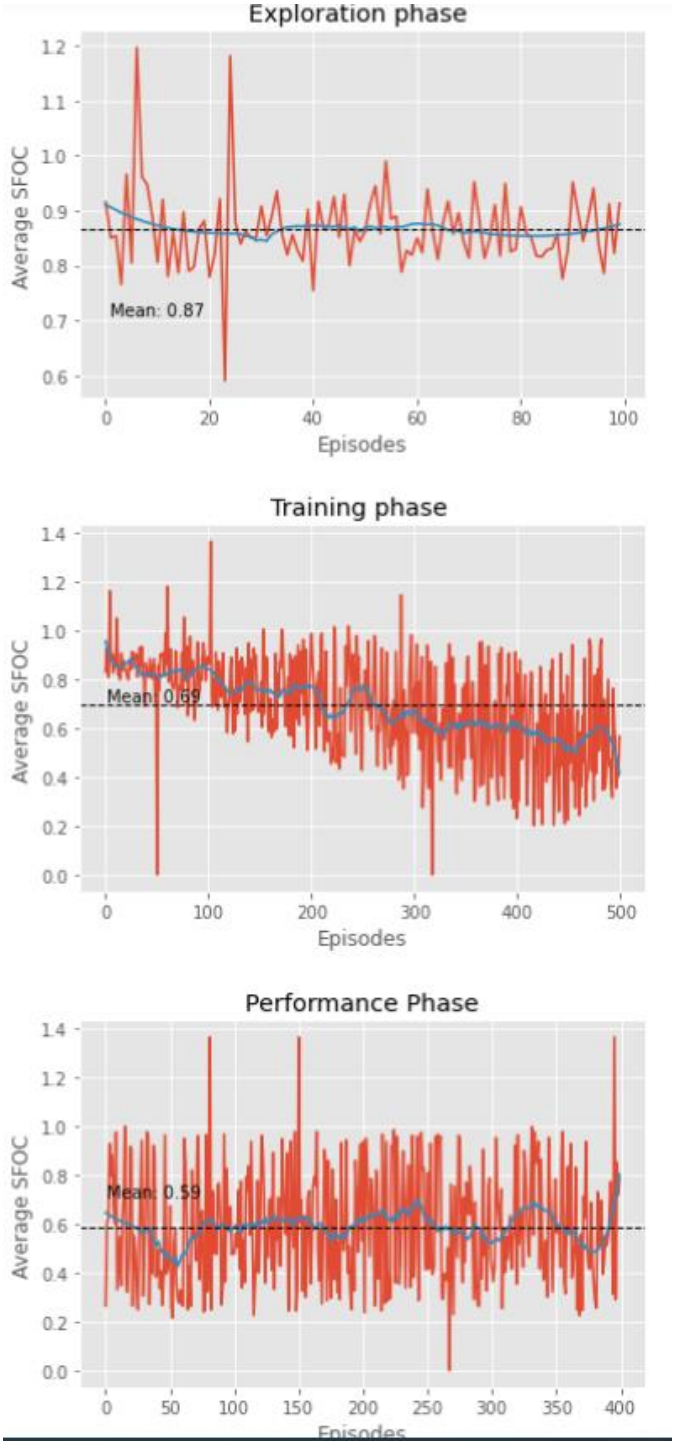
Figur 2: Reward Dsitribution



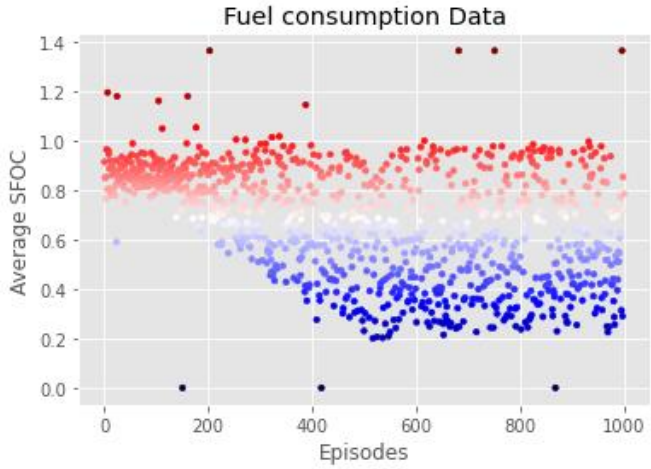
Figur 3:Density plot of training

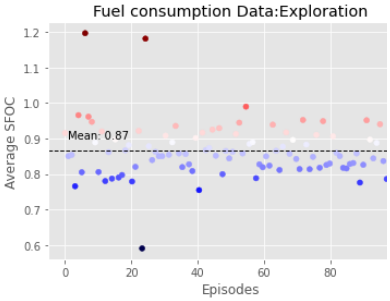


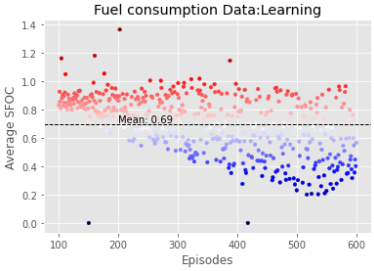
Figur 4: Reward Dsitribution

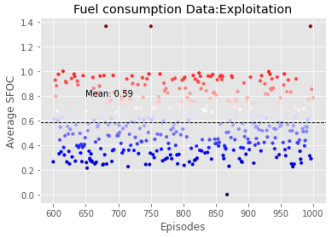

Exploration phase
Average SFOC
Mean: 0.87
Episodes
Training phase
Average SFOC
Mean: 0.69
Episodes
Performance Phase
Average SFOC
Mean: 0.59
Episodes

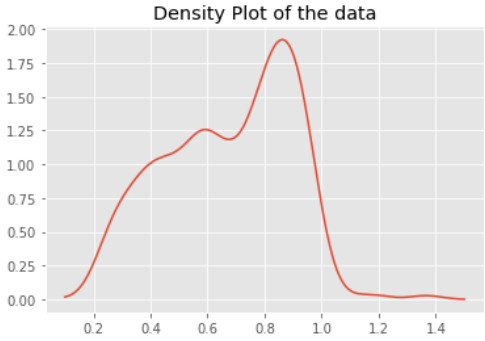
Figur 5:SFOC

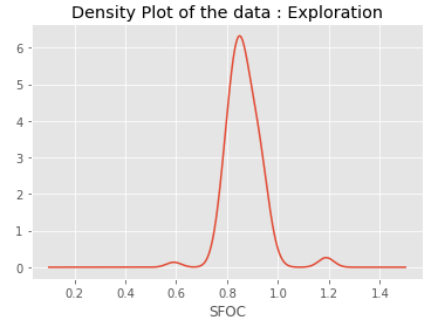

Fuel consumption Data
Average SFOC
Episodes

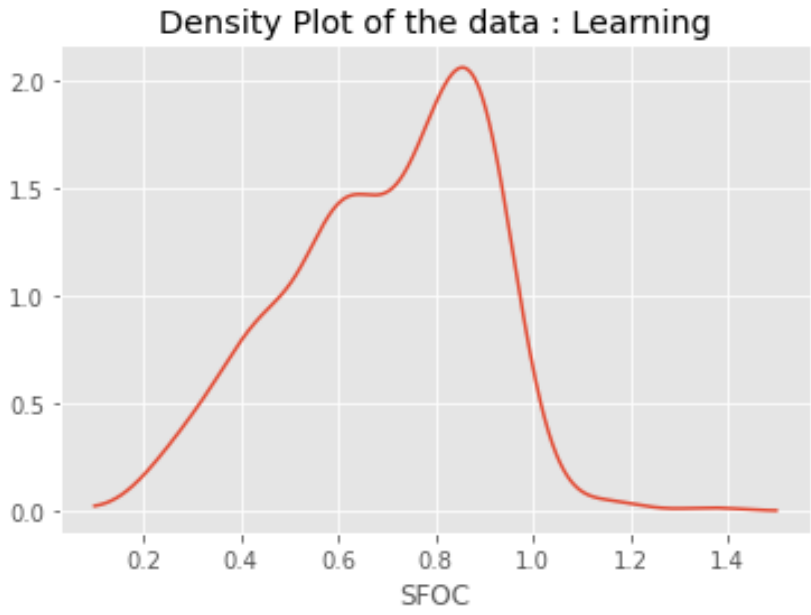

Fuel consumption Data:Exploration
Mean: 0.87
Average SFOC
Episodes

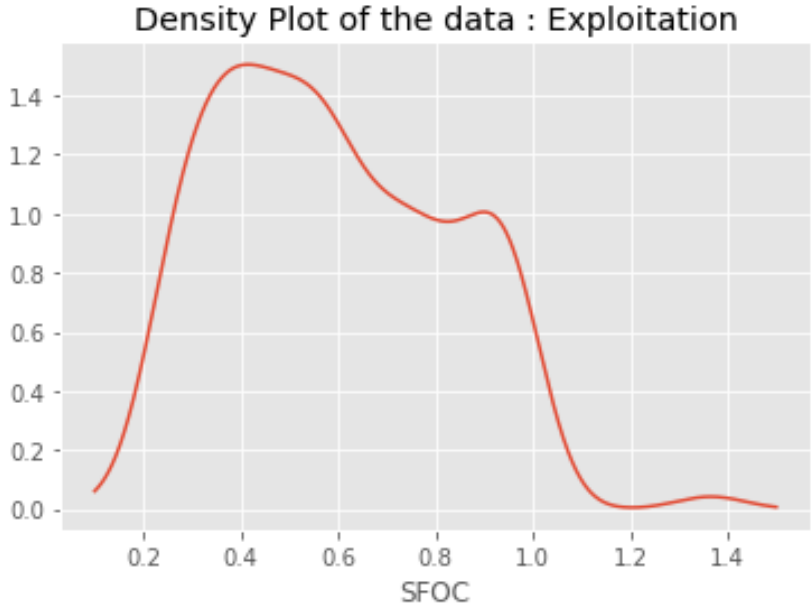

Fuel consumption Data:Learning
Mean: 0.69
Average SFOC
Episodes

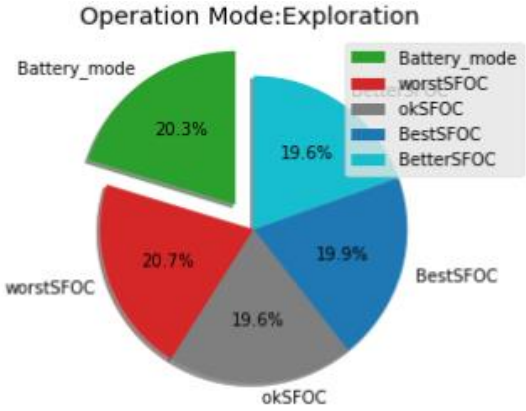

Fuel consumption Data:Exploitation
Mean: 0.59
Average SFOC
Episodes

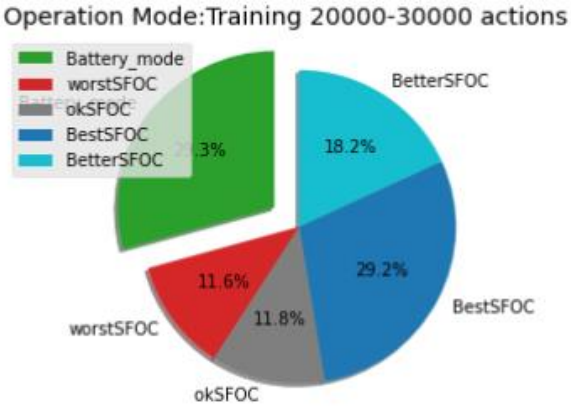

Density Plot of the data

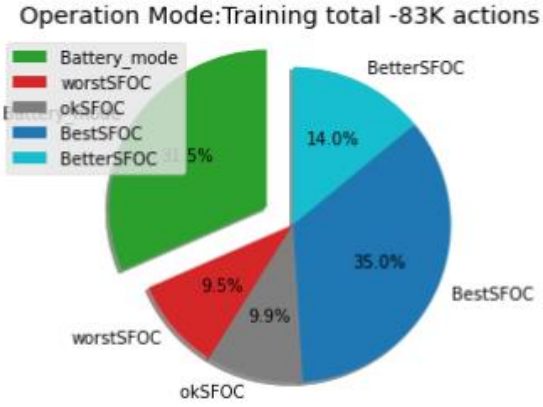

Density Plot of the data : Exploration
SFOC

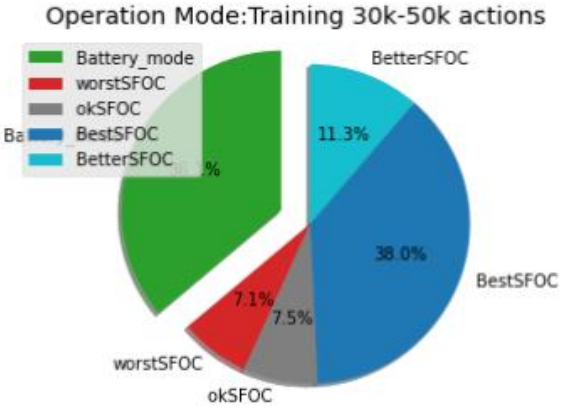

Density Plot of the data : Learning
2.0
1.5
1.0
0.5
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
SFOC

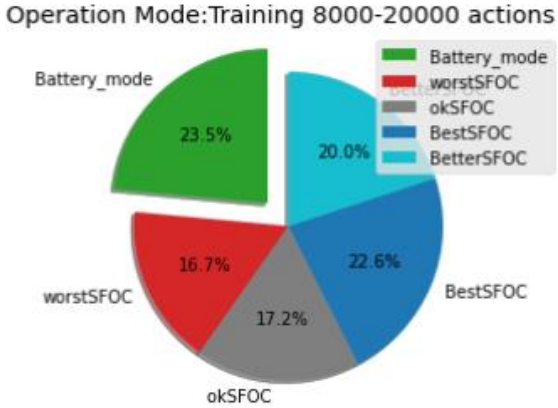

Density Plot of the data : Exploitation
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
SFOC

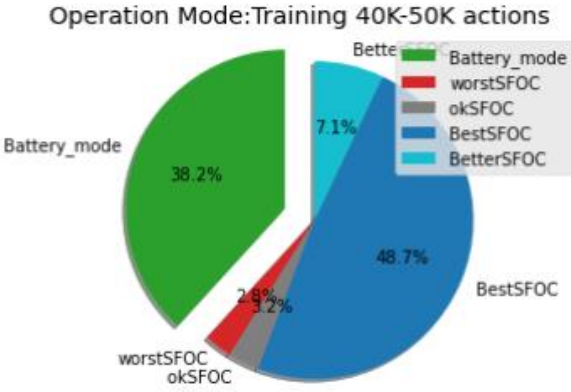








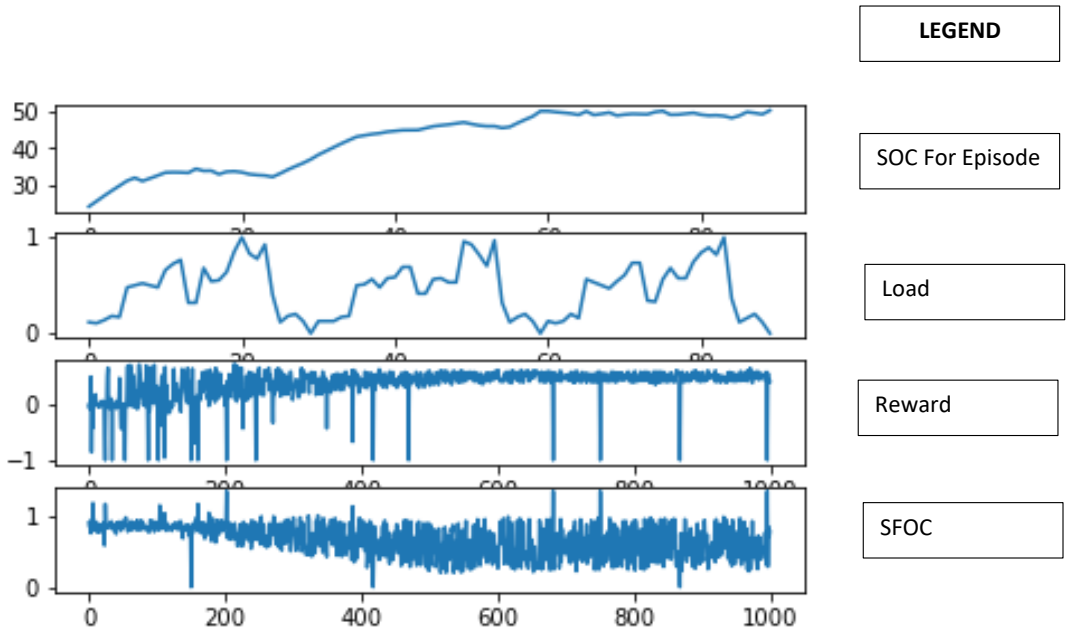




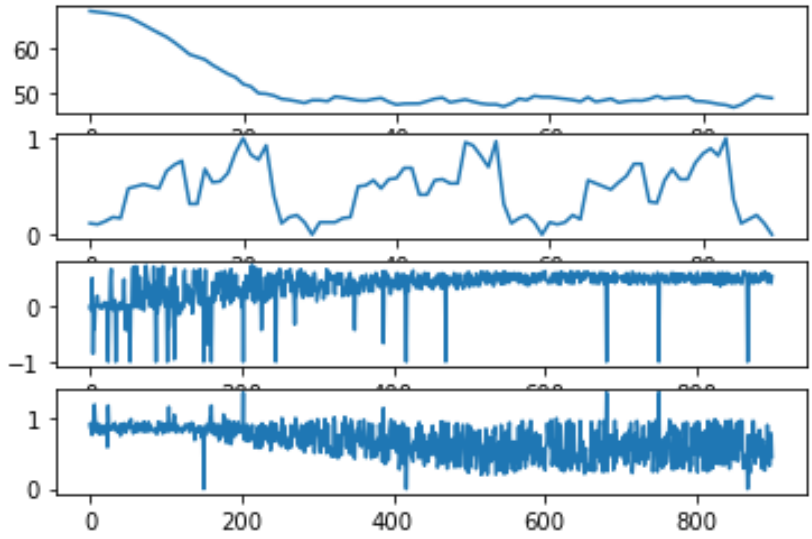
Figur 6: Actions

Operation Mode:Training 20000-30000 actions

Operation Mode:Training 50k-60k actions



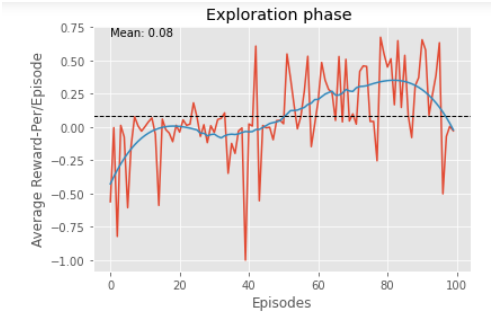
Figur 8: Episode number100 : From Top SOC , Load , Reward till episode , SFOC

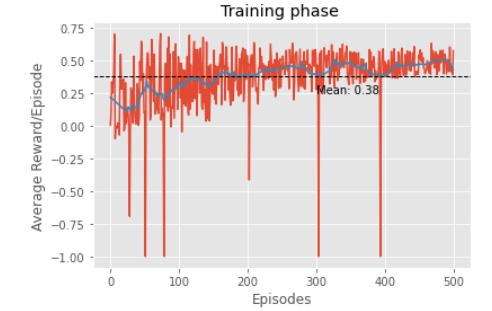


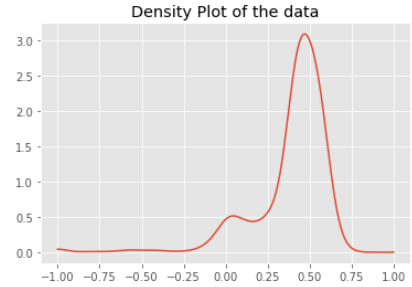
Figur 7: For episode 900

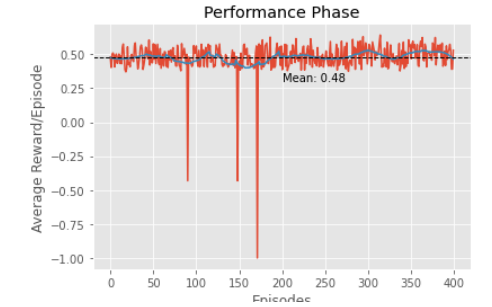
For full simulation result of 1000 soc : Check github repositry

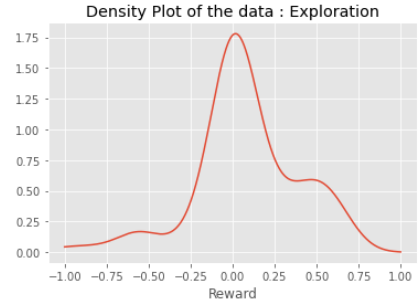
Experiment 1.2 : Tanh-Tanh-Linear Model

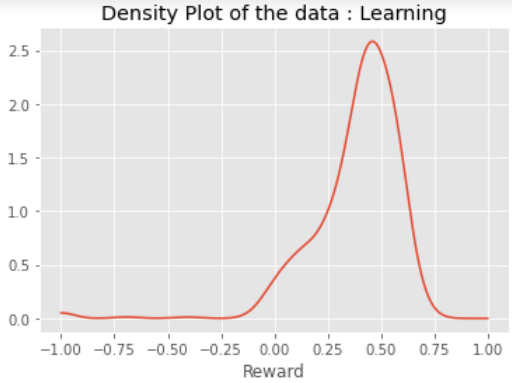


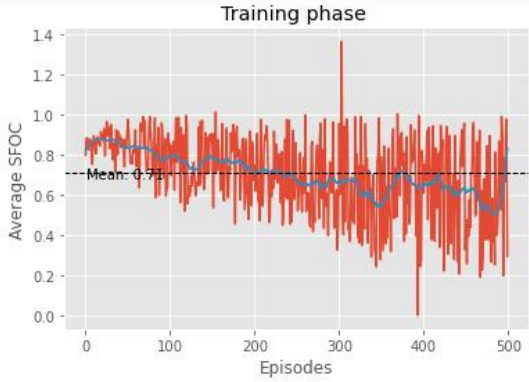


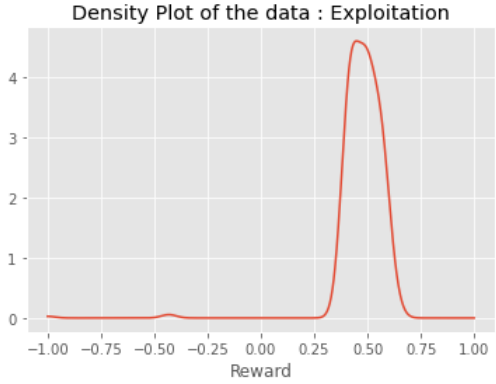




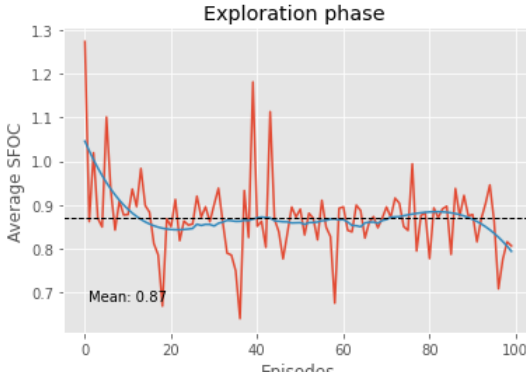


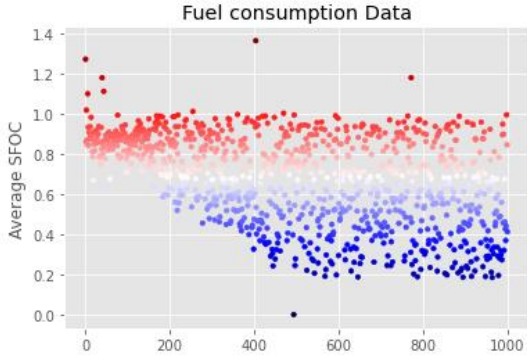

Density Plot of the data : Learning
Reward

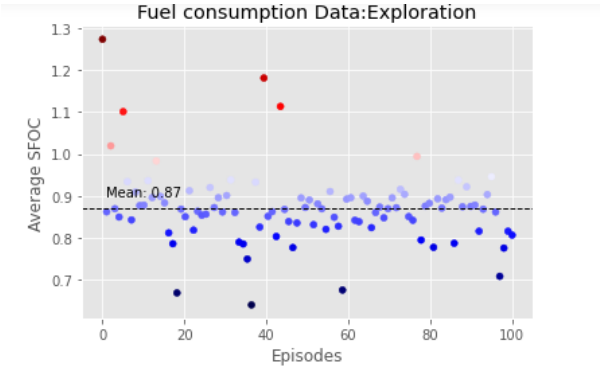

Training phase
Average SFOC
Mean: 0.71
Episodes

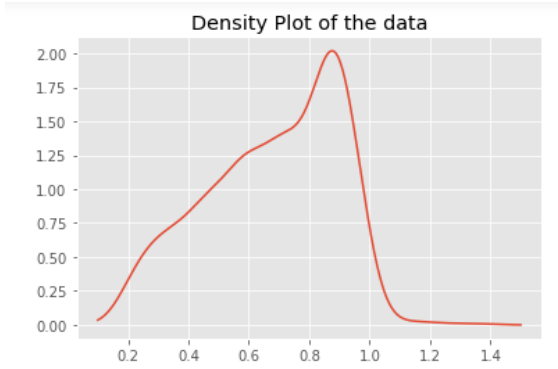

Density Plot of the data : Exploitation
Reward

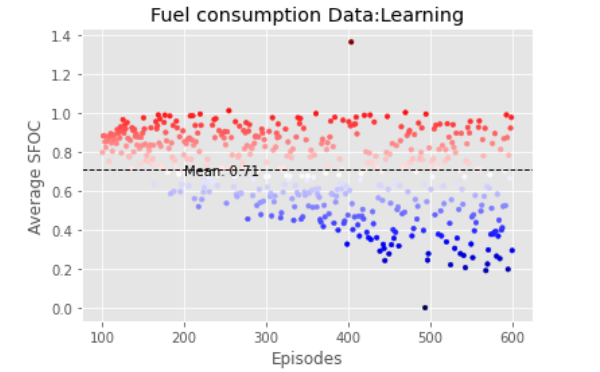

Performance Phase
Average SFOC
Mean: 0.59
Episodes

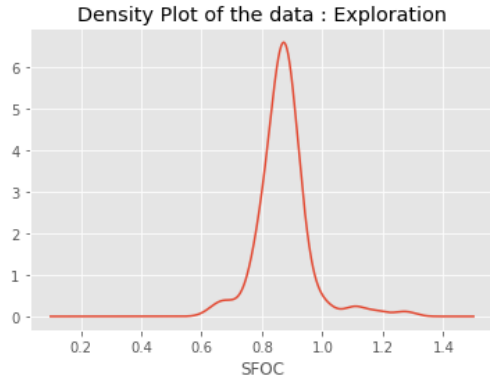

Exploration phase
Average SFOC
Mean: 0.87

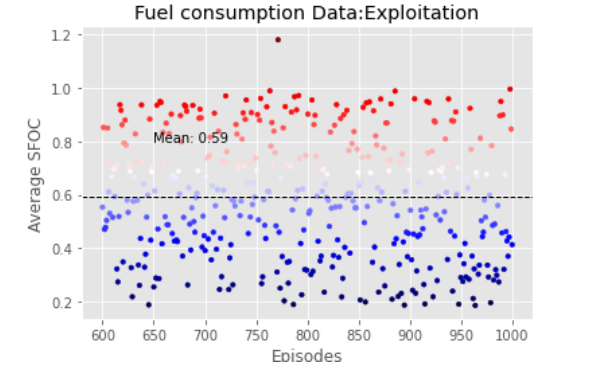

Fuel consumption Data
Average SFOC

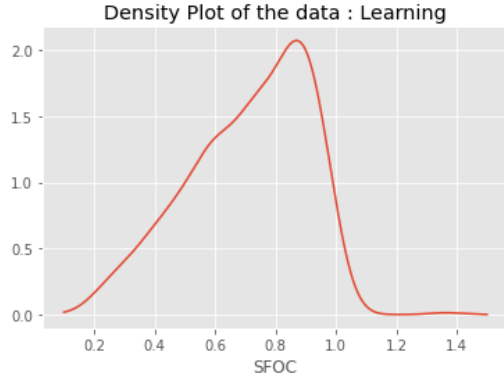

Fuel consumption Data:Exploration
Average SFOC
Episodes
Mean: 0.87

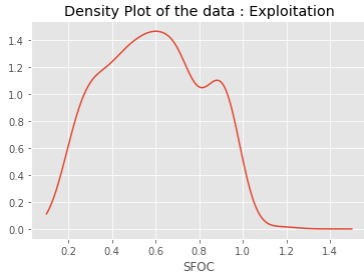

Density Plot of the data


Fuel consumption Data:Learning
Average SFOC
Episodes
Mean: 0.71

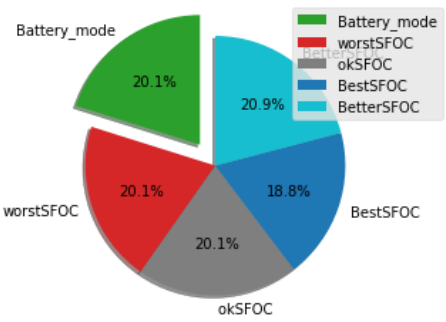

Density Plot of the data : Exploration
SFOC


Fuel consumption Data:Exploitation
Average SFOC
Episodes
Mean: 0.59

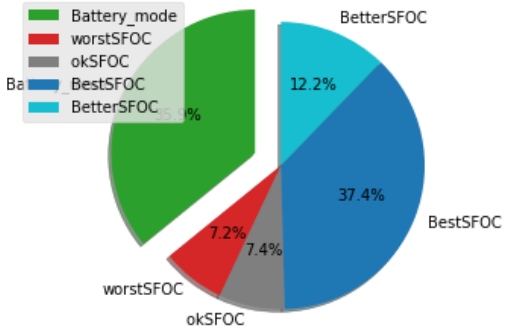

Density Plot of the data : Learning
SFOC


Density Plot of the data : Exploitation
SFOC

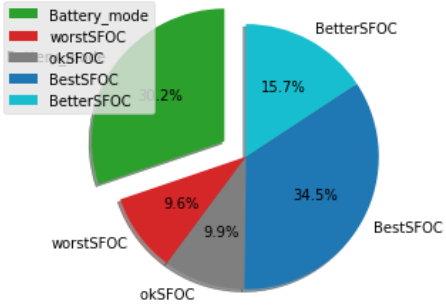
Operation Mode:Exploration



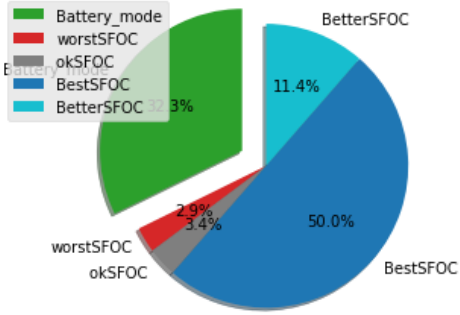
Operation Mode:Training 30k-50k actions



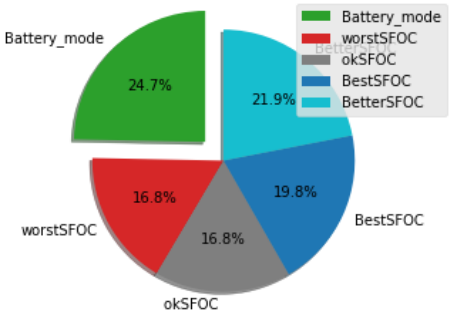
Operation Mode:Training total -83K actions



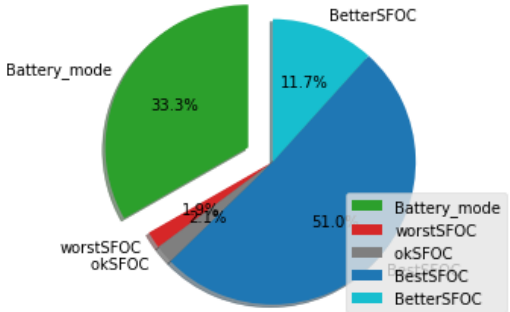
Operation Mode:Training 40K-50K actions



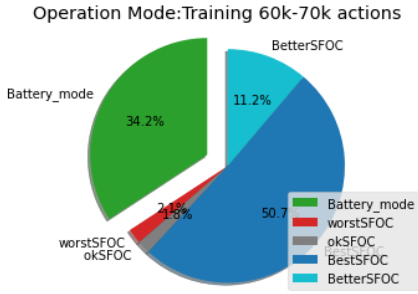
Operation Mode:Training 8000-20000 actions

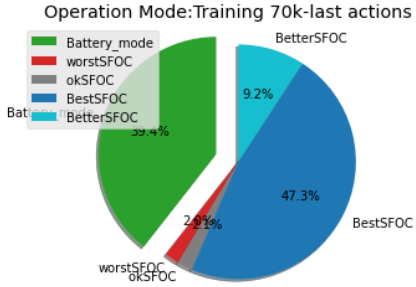


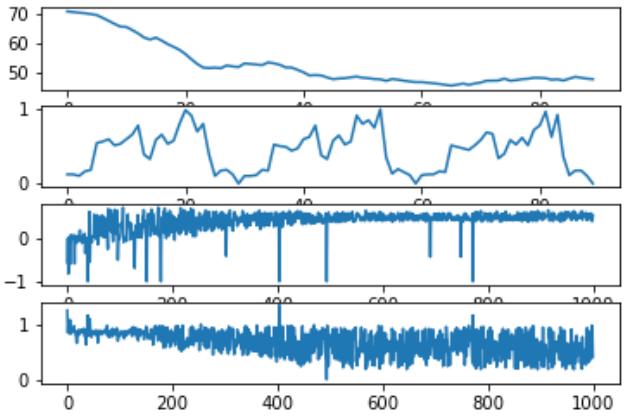
Operation Mode:Training 50k-60k actions



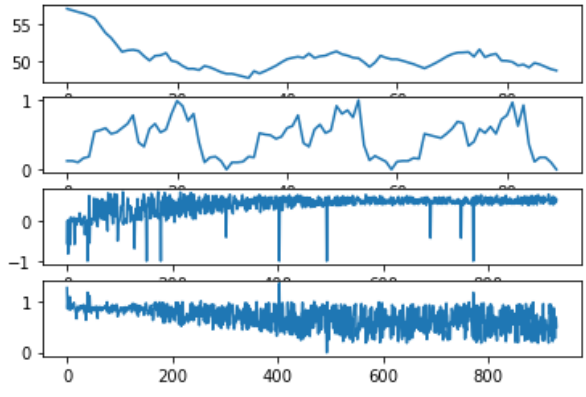
Operation Mode:Training 20000-30000 actions







Figur 2: Episode 1000



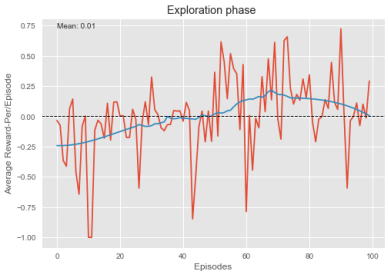
Figur 1:Episode 931

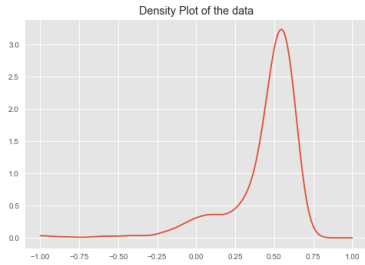
For full simulation result see Github Repository

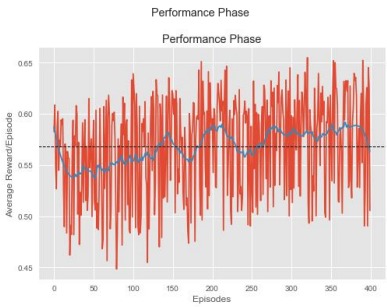
Appendix B

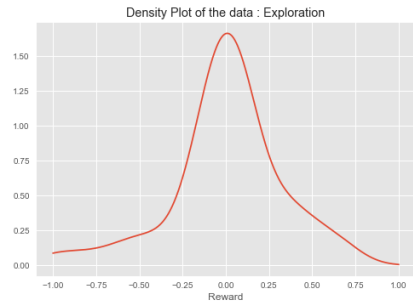
Appendix

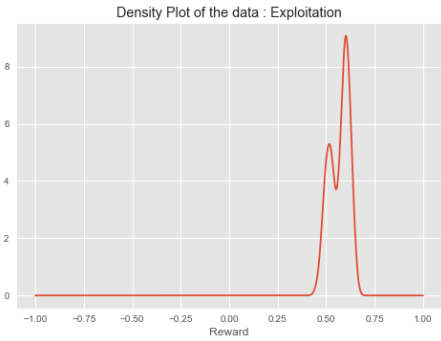
Experiment 2.1 : Reduced Storage – Tanh – Relu – Linear

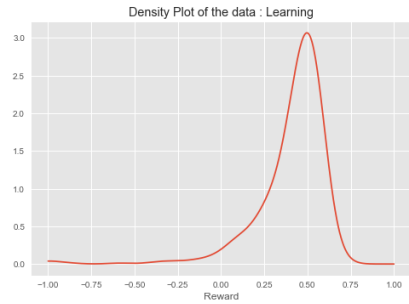


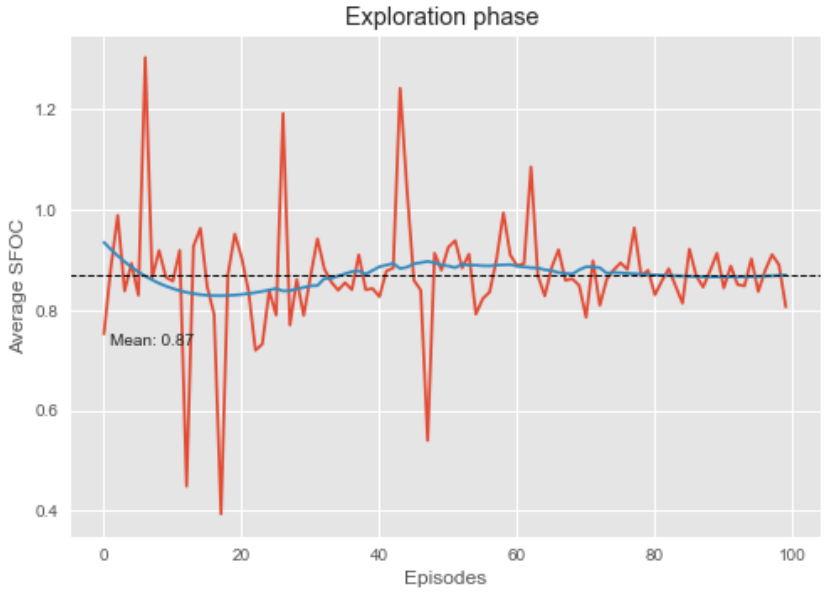




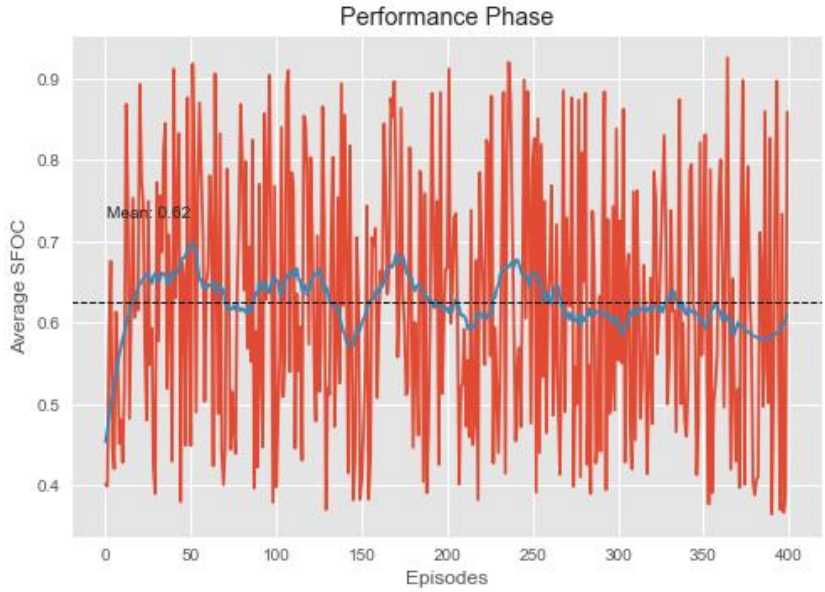


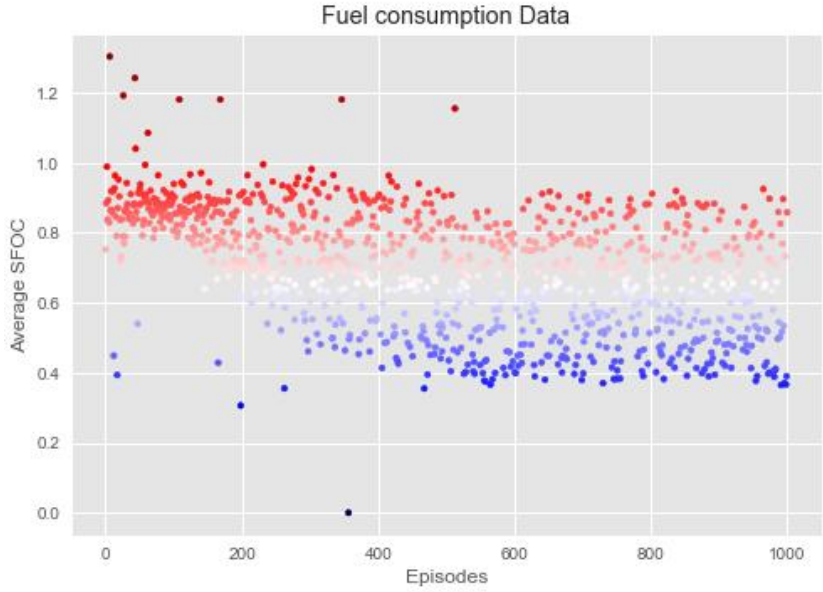


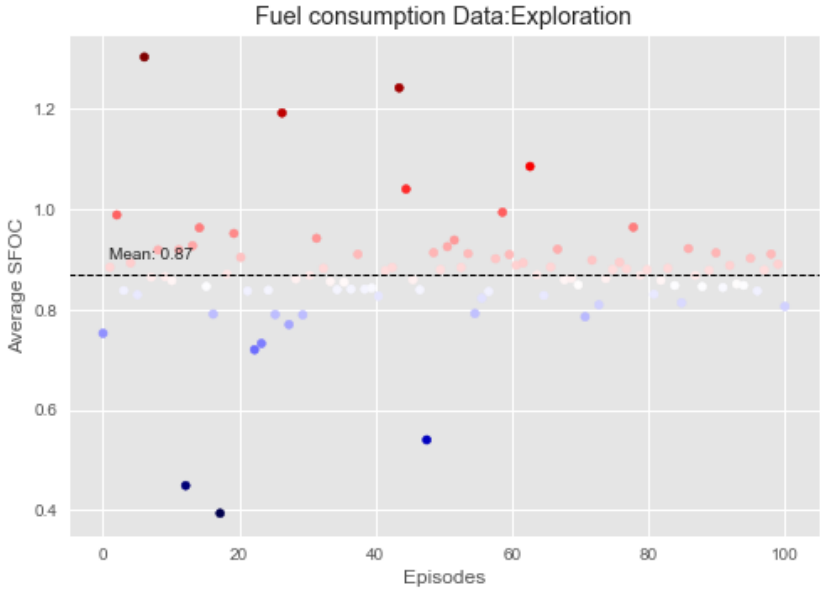




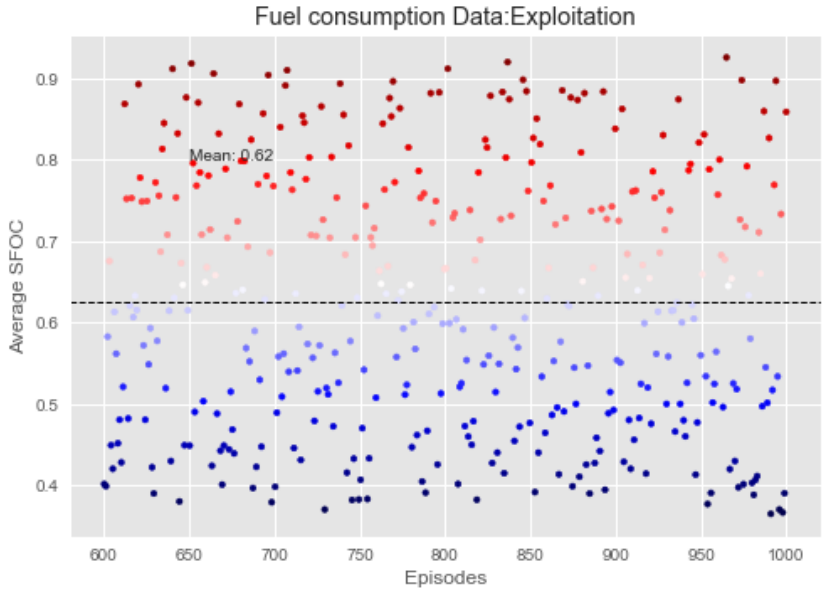
Performance Phase

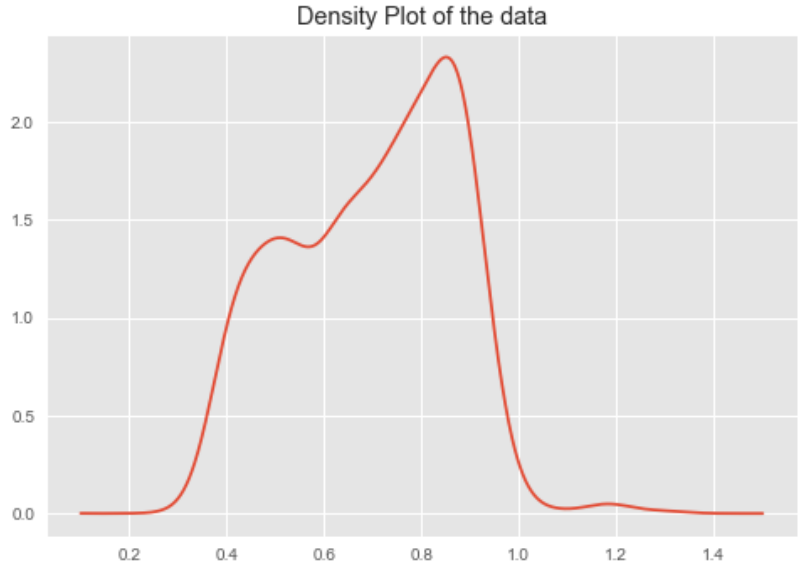

Performance Phase
Mean: 0.62
Average SFOC
Episodes
0.4
0.5
0.6
0.7
0.8
0.9
0
50
100
150
200
250
300
350
400

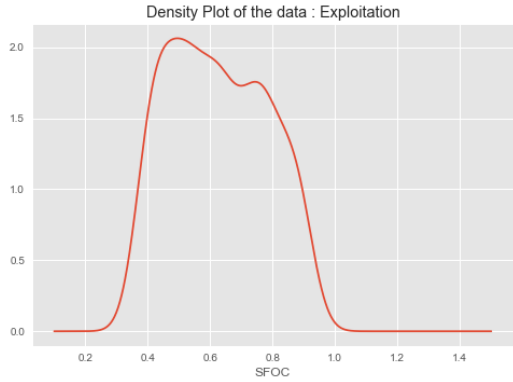

Fuel consumption Data
Average SFOC
Episodes
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0
200
400
600
800
1000

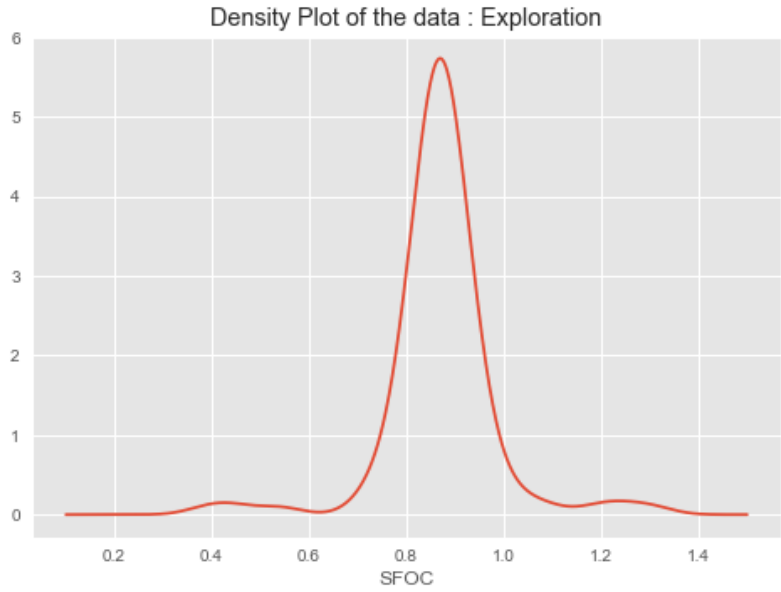

Fuel consumption Data:Exploration
Average SFOC
Episodes
Mean: 0.87
1.2
1.0
0.8
0.6
0.4
0
20
40
60
80
100


Fuel consumption Data:Learning
Average SFOC
Episodes
Mean: 0.72
1.2
1.0
0.8
0.6
0.4
0.2
0.0
100
200
300
400
500
600

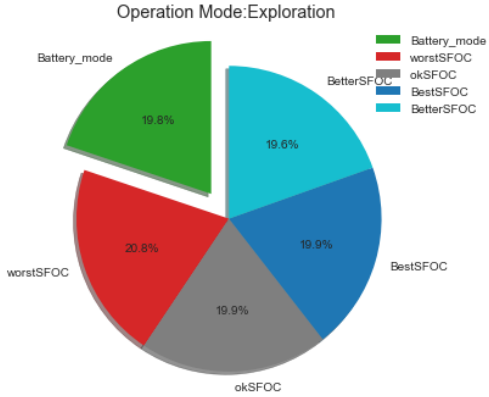

Fuel consumption Data:Exploitation
Mean: 0.62
Average SFOC
Episodes
0.9
0.8
0.7
0.6
0.5
0.4
600
650
700
750
800
850
900
950
1000

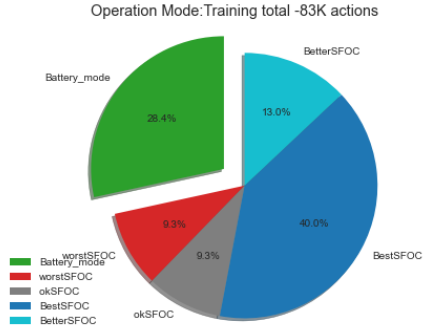

Density Plot of the data
2.0
1.5
1.0
0.5
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

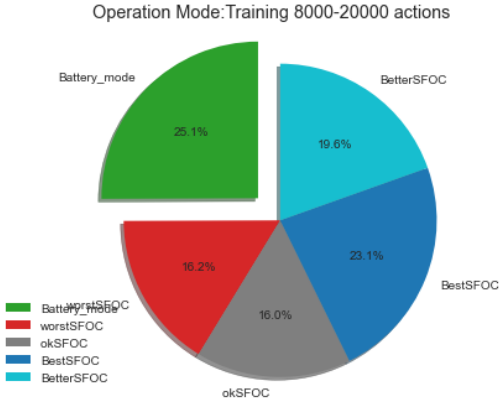

Density Plot of the data : Exploitation
2.0
1.5
1.0
0.5
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
SFOC

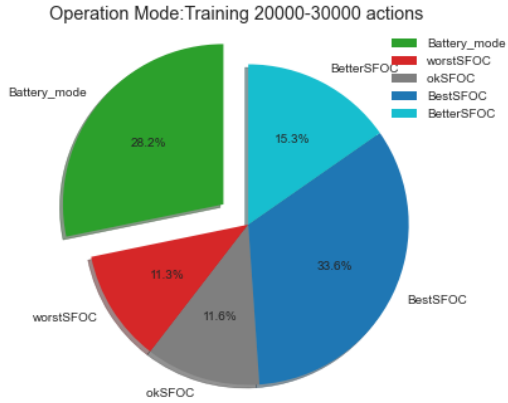

Density Plot of the data : Exploration
6
5
4
3
2
1
0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
SFOC

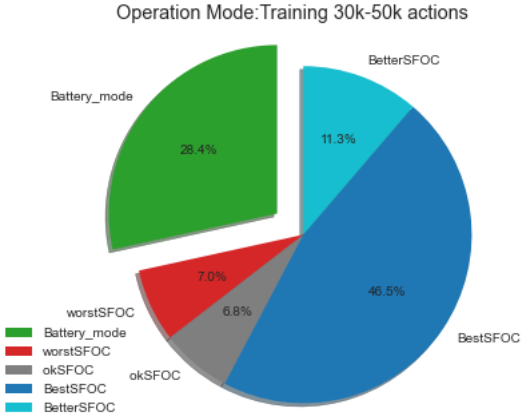

Density Plot of the data : Learning
2.5
2.0
1.5
1.0
0.5
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
SFOC

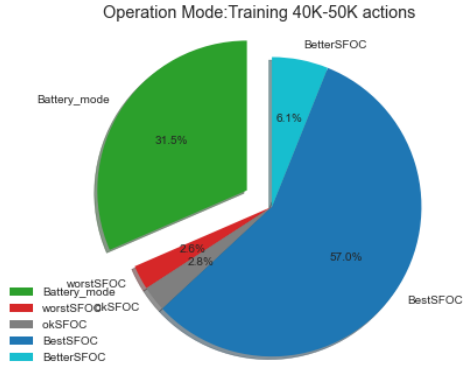

Operation Mode:Exploration
Battery_mode
worstSFOC
okSFOC
BestSFOC
BetterSFOC
19.8%
19.6%
20.8%
19.9%
19.9%

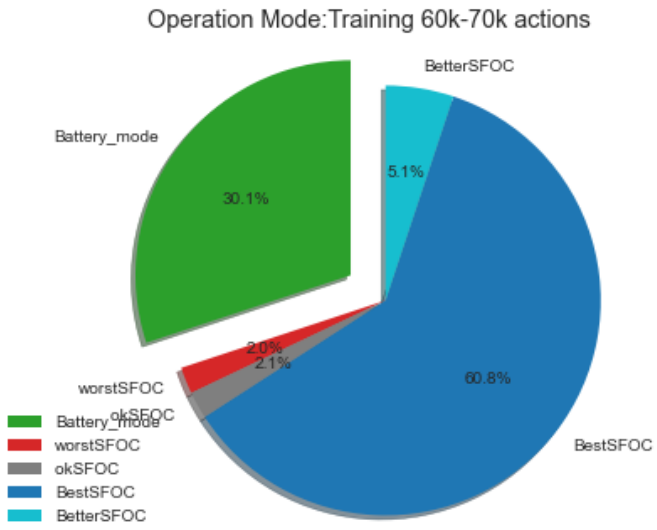

Operation Mode:Training total -83K actions
Battery_mode
BetterSFOC
28.4%
13.0%
9.3%
40.0%
9.3%
worstSFOC
okSFOC
BestSFOC

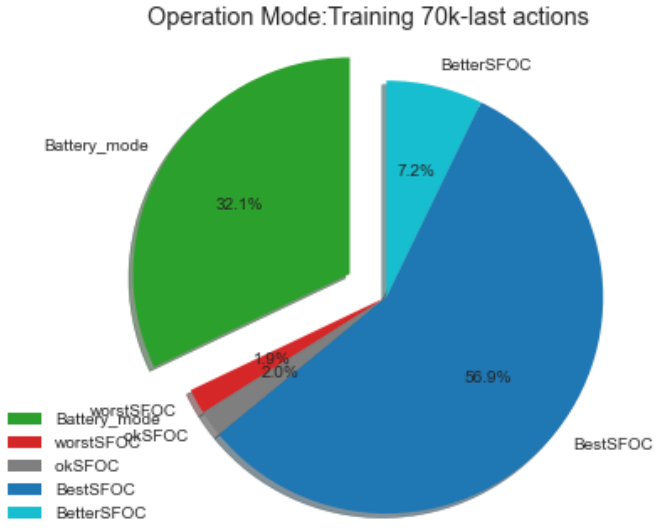

Operation Mode:Training 8000-20000 actions
Battery_mode
BetterSFOC
25.1%
19.6%
16.2%
23.1%
16.0%
BestSFOC
okSFOC


Operation Mode:Training 20000-30000 actions
Battery_mode
worstSFOC
okSFOC
BestSFOC
BetterSFOC
28.2%
15.3%
11.3%
33.6%
11.6%
worstSFOC
okSFOC
BestSFOC

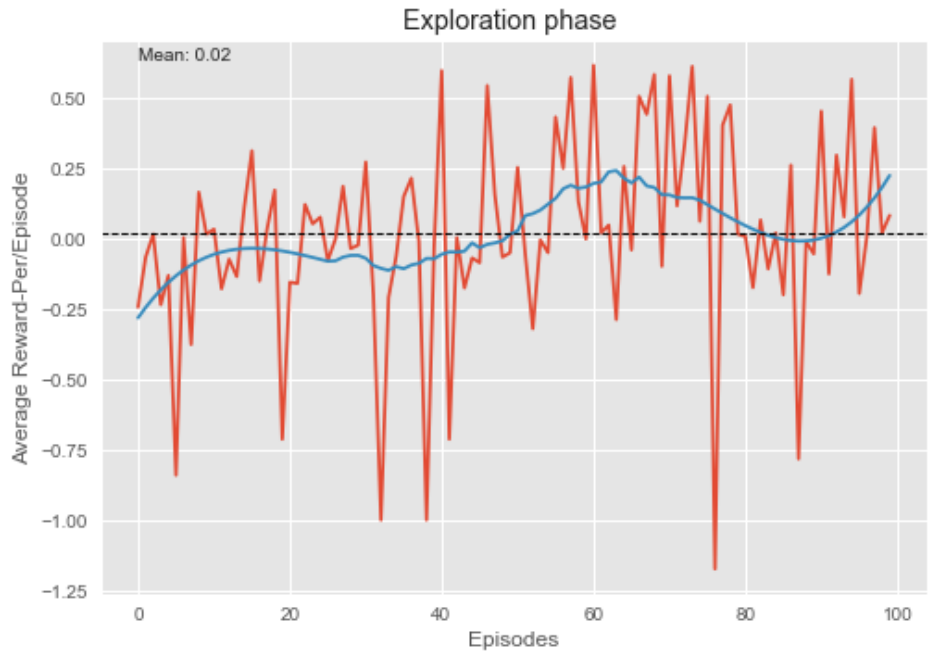

Operation Mode:Training 30k-50k actions
Battery_mode
BetterSFOC
28.4%
11.3%
7.0%
46.5%
6.8%
worstSFOC
okSFOC
BestSFOC


Operation Mode:Training 40K-50K actions
Battery_mode
BetterSFOC
31.5%
6.1%
2.6%
2.8%
57.0%
worstSFOC
okSFOC
BestSFOC

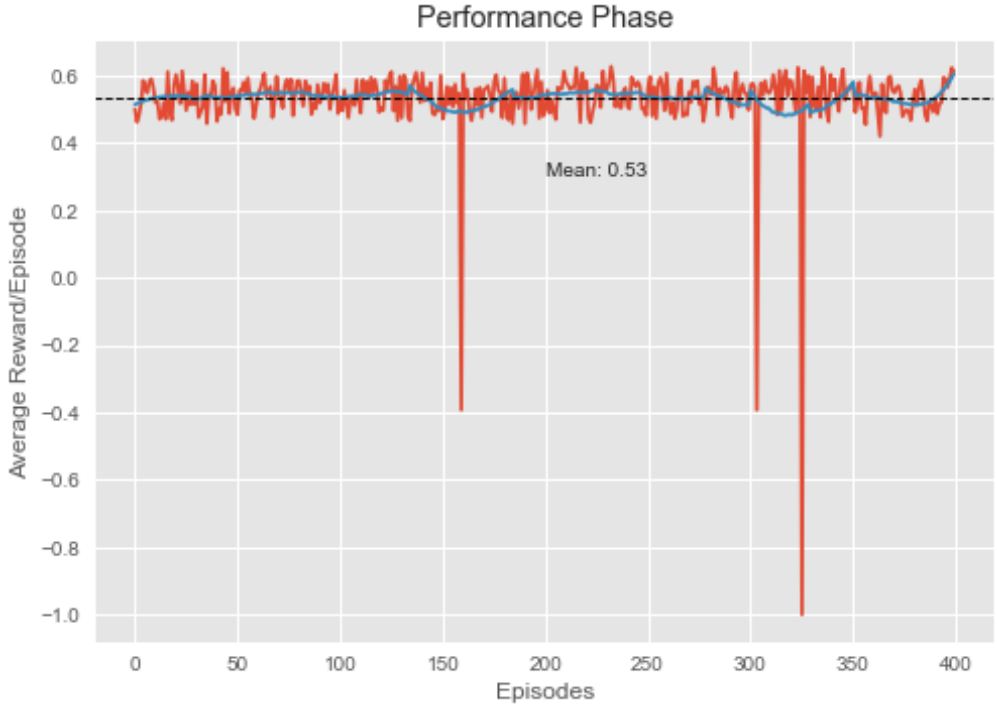

Operation Mode:Training 60k-70k actions
BetterSFOC
Battery_mode
30.1%
5.1%
2.0%
2.1%
worstSFOC
okSFOC
60.8%
BestSFOC
Battery_mode
worstSFOC
okSFOC
BestSFOC
BetterSFOC


Operation Mode:Training 70k-last actions
BetterSFOC
Battery_mode
7.2%
32.1%
1.9%
2.0%
56.9%
worstSFOC
okSFOC
BestSFOC
Battery_mode
worstSFOC
okSFOC
BestSFOC
BetterSFOC

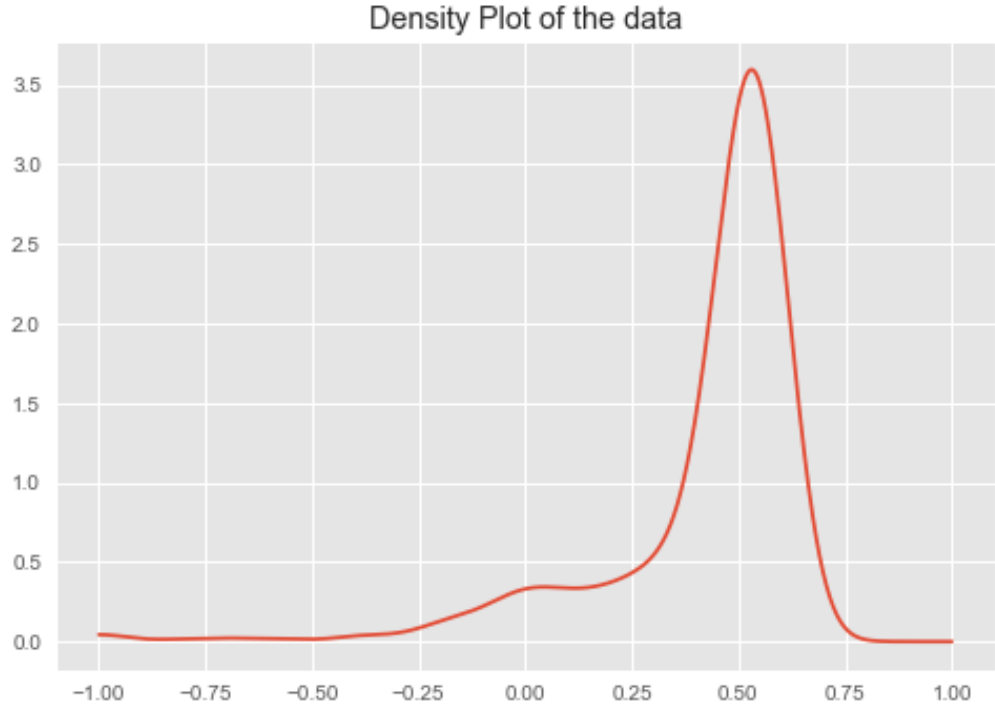
Experiment 2.2 Reduced Storage : Tanh-Tanh- Linear

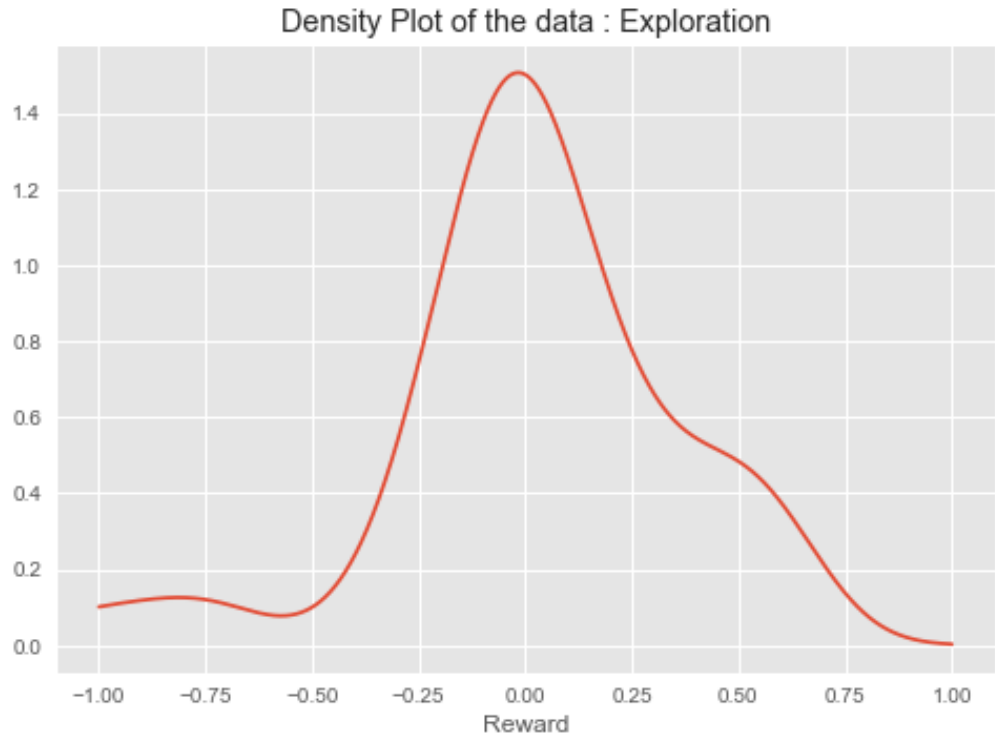


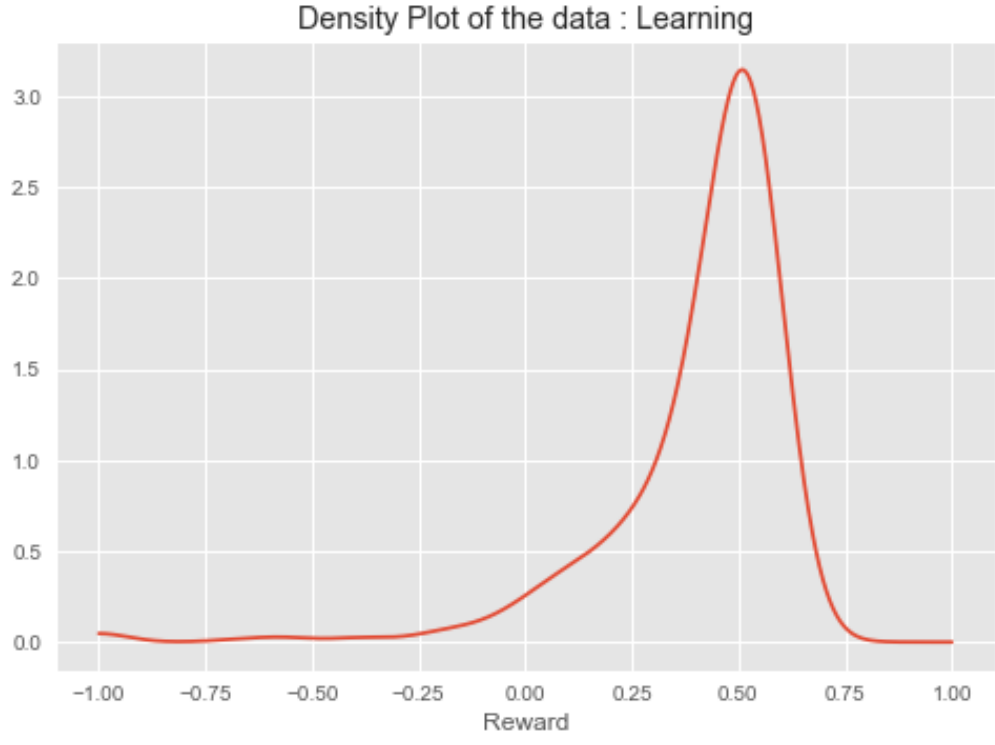

Reward data
Average Reward/Episode
Episodes
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.25
0
200
400
600
800
1000

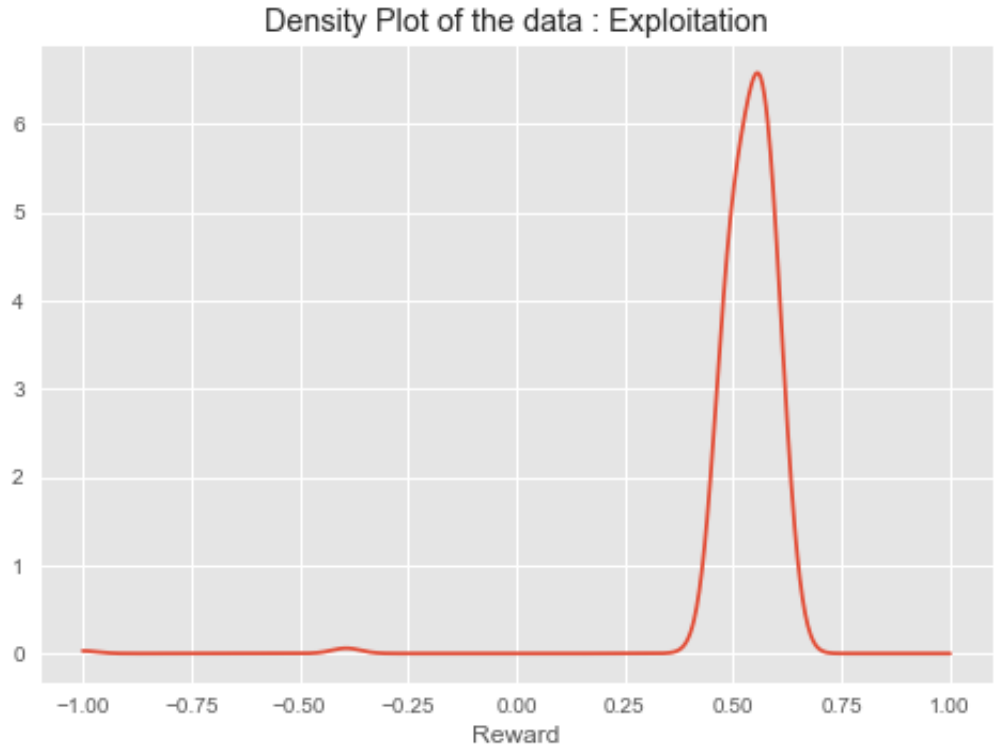

Performance Phase
Mean: 0.53
Average Reward/Episode
Episodes
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
0
50
100
150
200
250
300
350
400

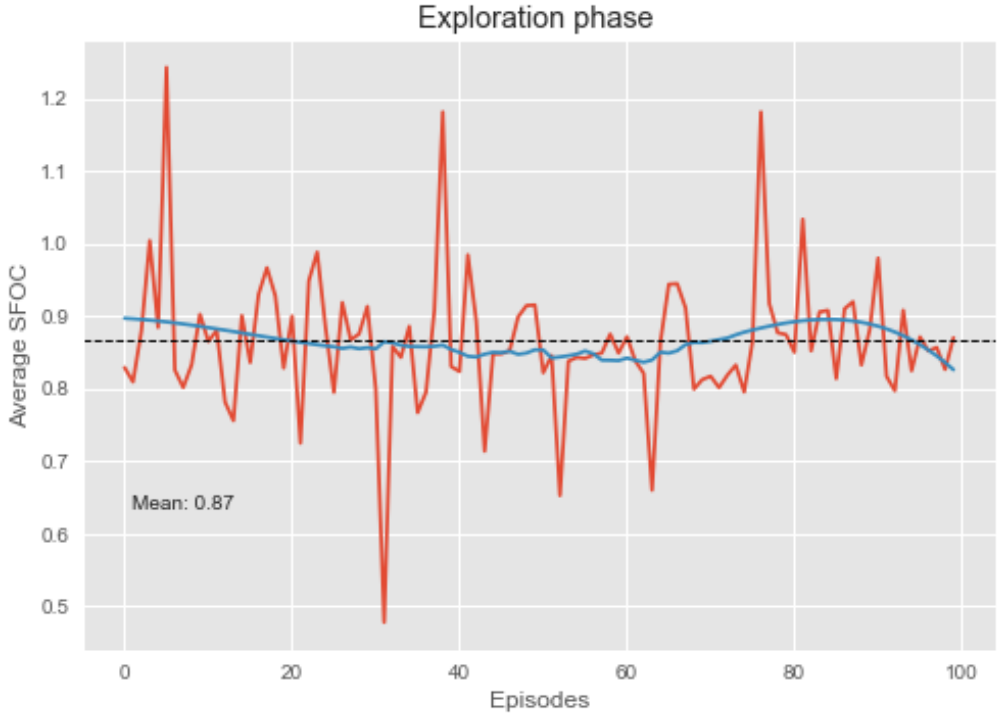

Reward data
Average Reward/Episode
Episodes
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.25
0
200
400
600
800
1000

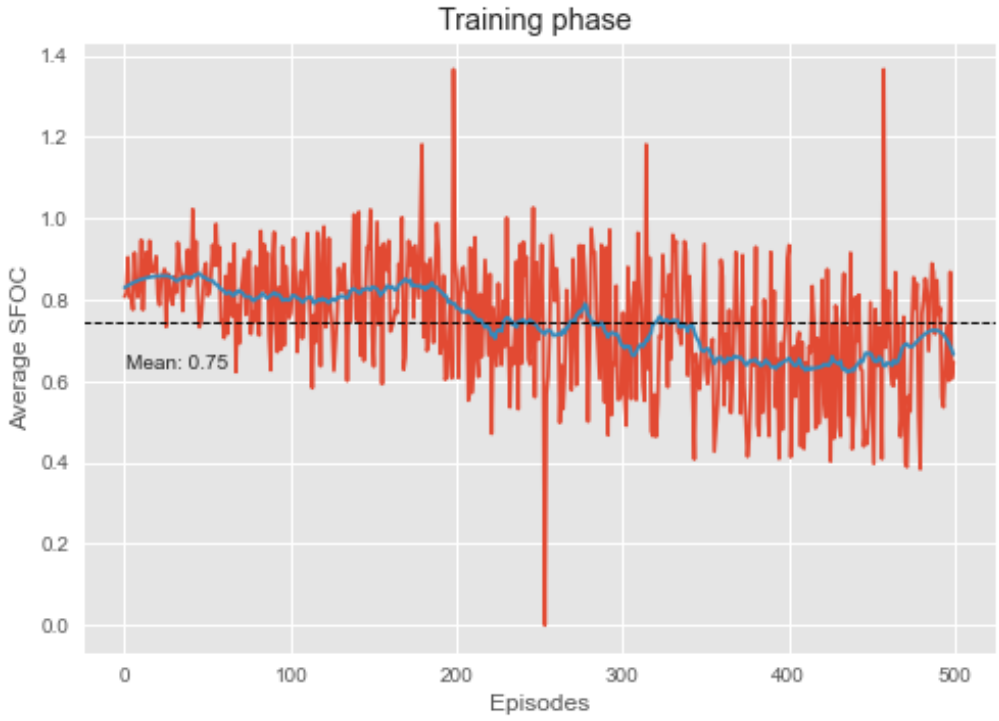

Density Plot of the data
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

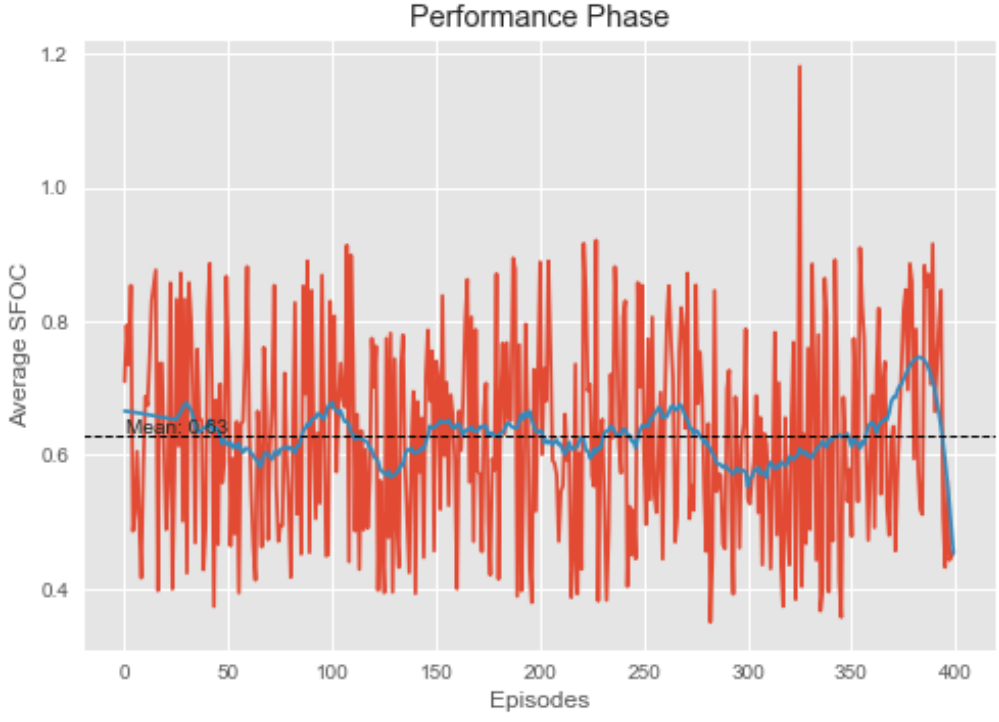

Density Plot of the data : Exploration
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
Reward

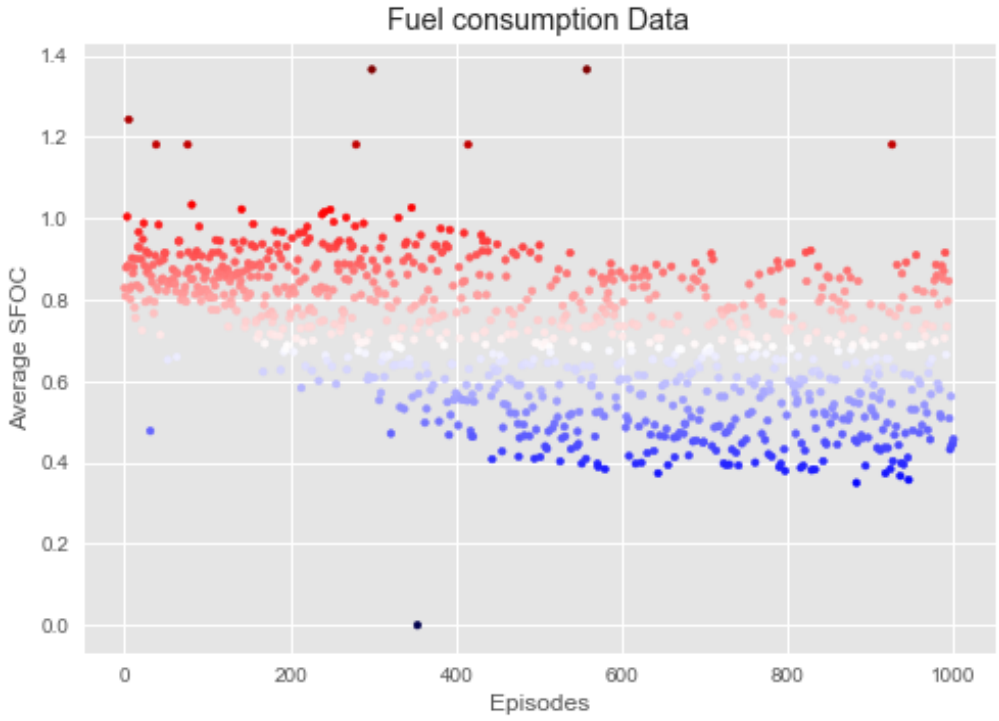

Density Plot of the data : Learning
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
Reward

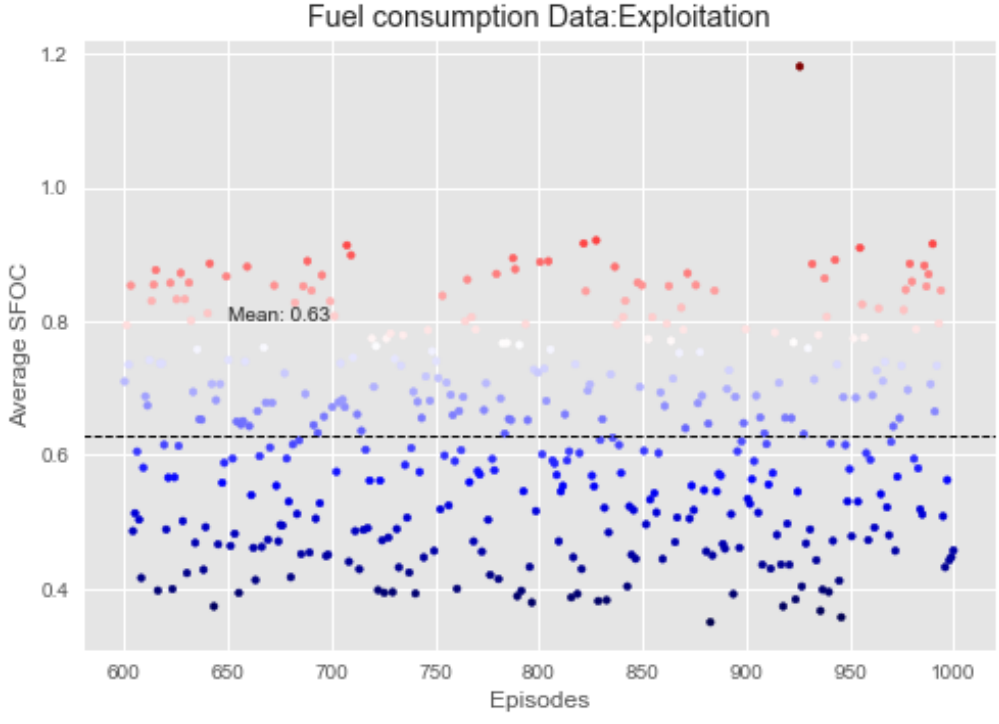

Density Plot of the data : Exploitation
6
5
4
3
2
1
0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
Reward

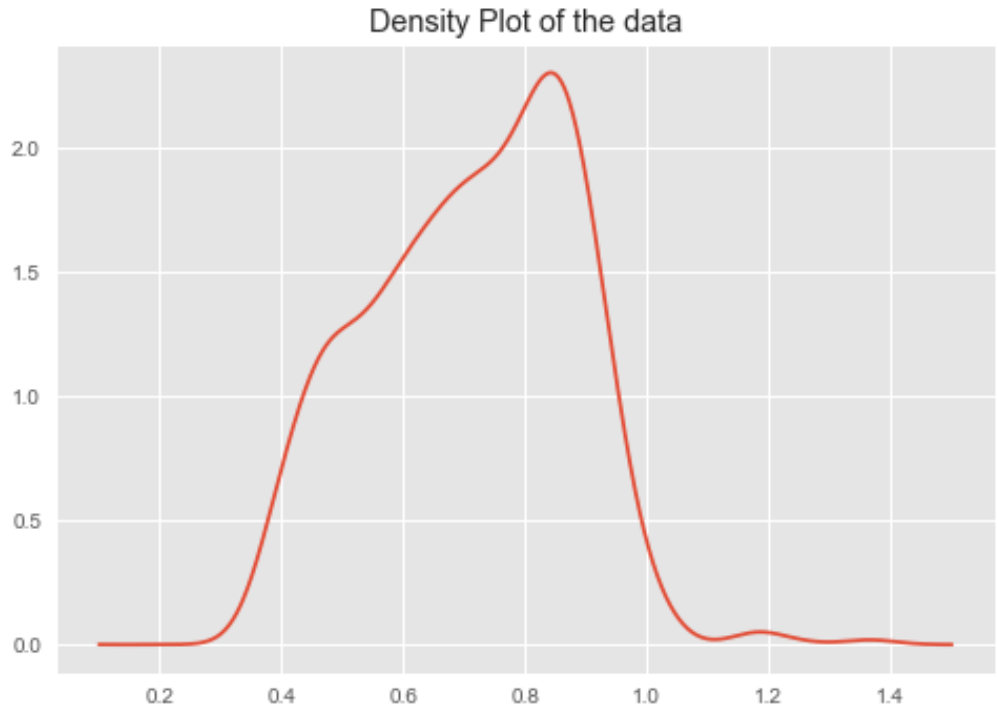

Exploration phase
Average SFOC
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
Mean: 0.87
0
20
40
60
80
100
Episodes

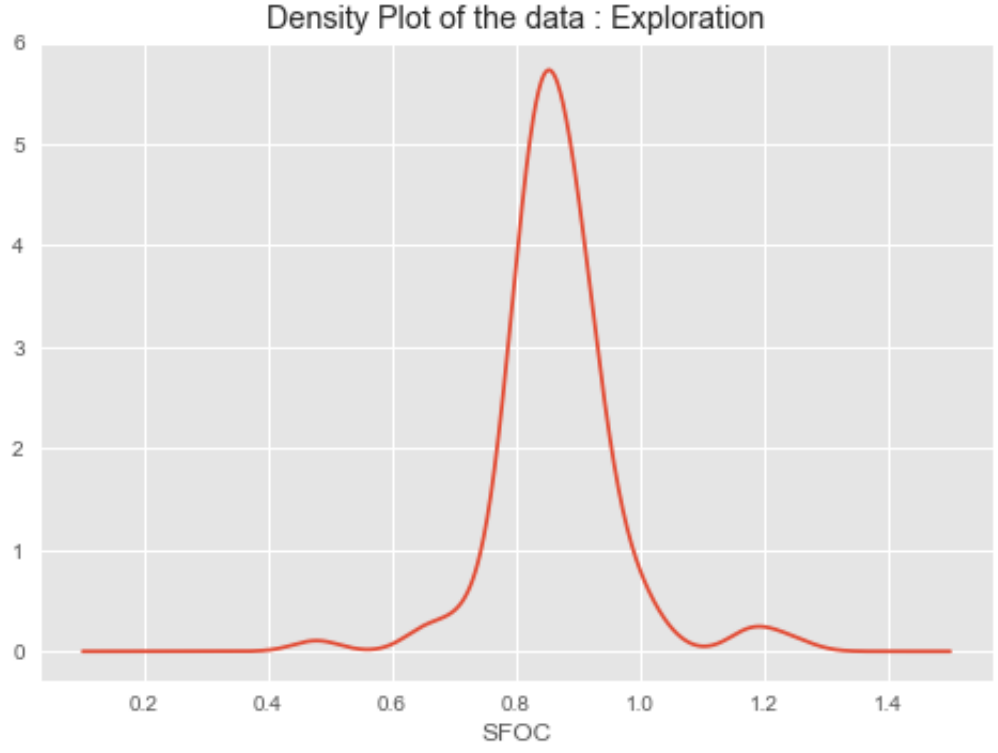

Training phase
Average SFOC
Episodes
Mean: 0.75
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
0
100
200
300
400
500


Performance Phase
Average SFOC
Episodes
Mean: 0.63
0.4
0.6
0.8
1.0
1.2
0
50
100
150
200
250
300
350
400

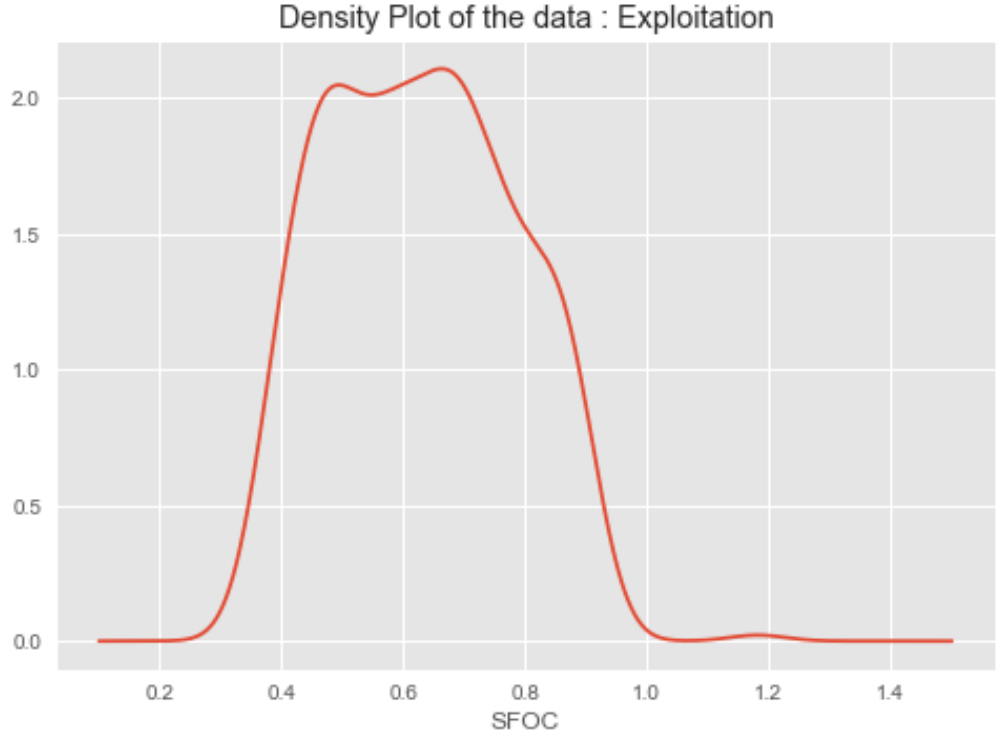

Fuel consumption Data
Average SFOC
Episodes
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000

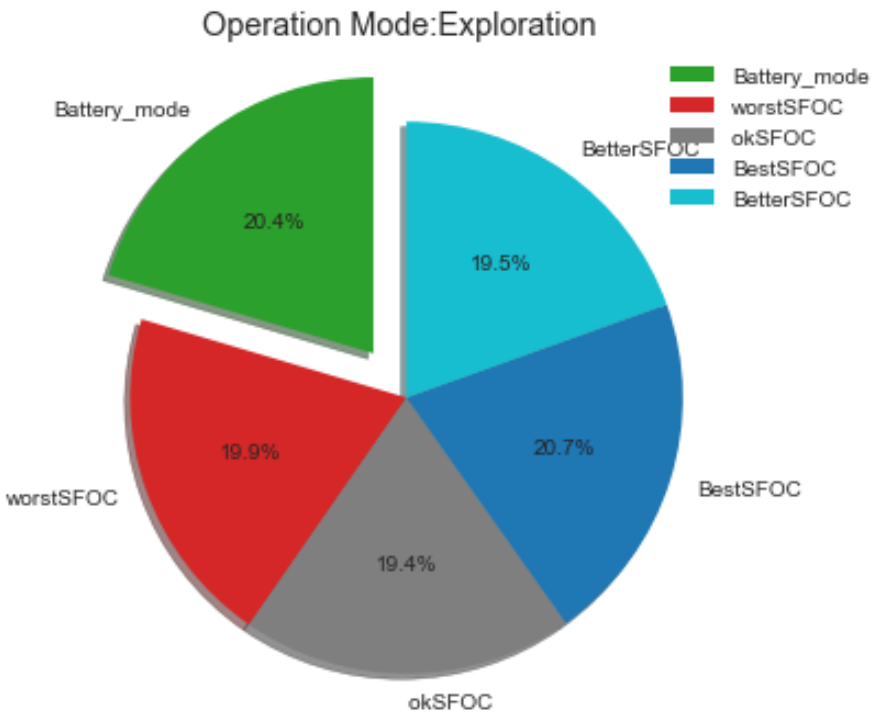

Fuel consumption Data:Exploitation
Average SFOC
Episodes
Mean: 0.63
1.2
1.0
0.8
0.6
0.4
600
650
700
750
800
850
900
950
1000

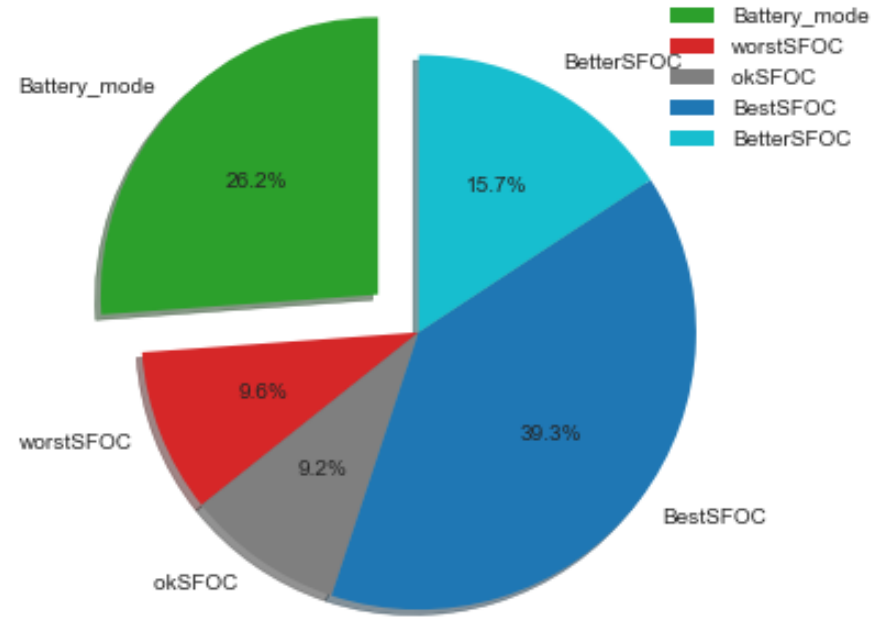

Density Plot of the data
2.0
1.5
1.0
0.5
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4


Density Plot of the data : Exploration
6
5
4
3
2
1
0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
SFOC

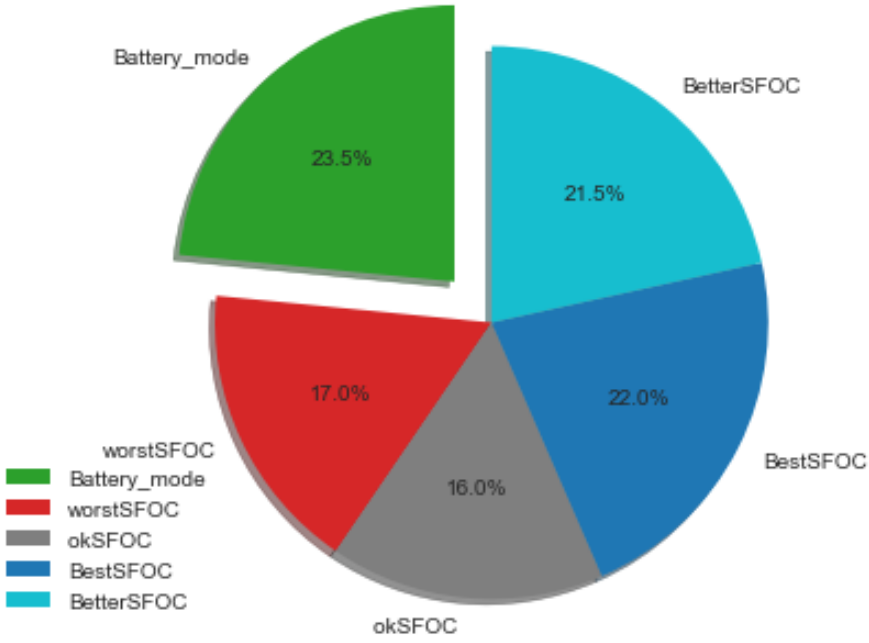

Density Plot of the data : Learning
2.5
2.0
1.5
1.0
0.5
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
SFOC


Density Plot of the data : Exploitation
2.0
1.5
1.0
0.5
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
SFOC

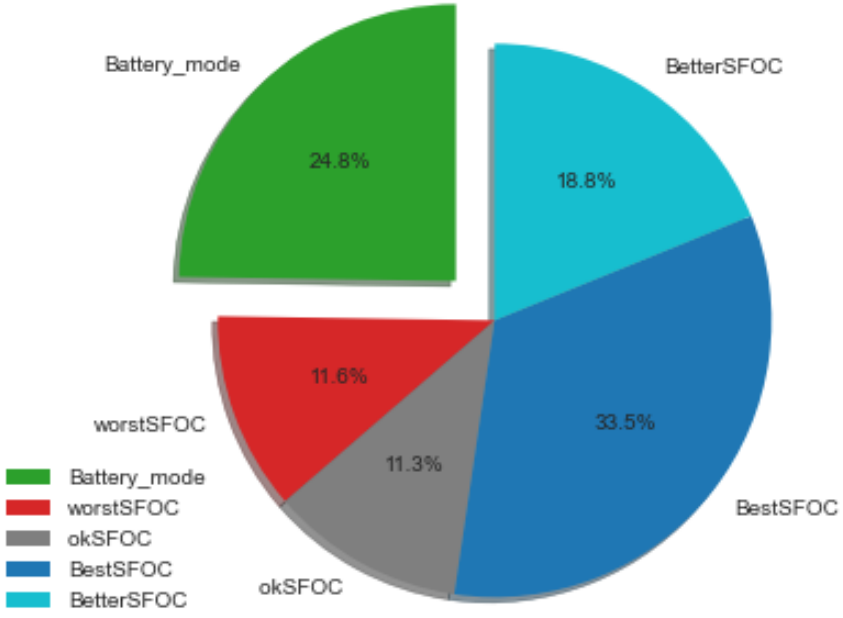

Operation Mode:Exploration
Battery_mode
worstSFOC
okSFOC
BestSFOC
BetterSFOC
Battery_mode
BetterSFOC
20.4%
19.5%
19.9%
20.7%
worstSFOC
BestSFOC
19.4%
okSFOC

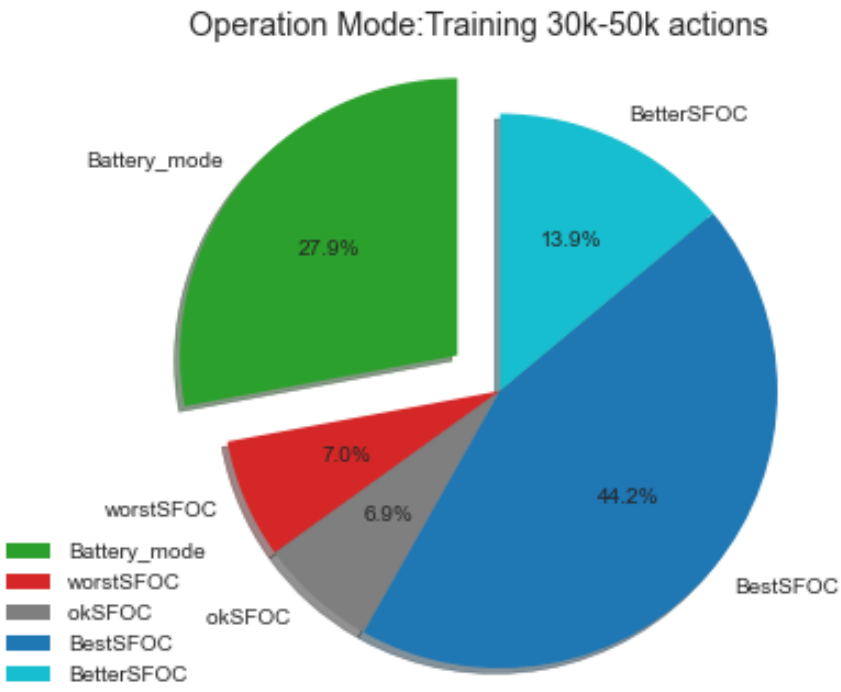

Operation Mode:Training total -83K actions
Battery_mode
worstSFOC
okSFOC
BestSFOC
BetterSFOC
Battery_mode
BetterSFOC
26.2%
15.7%
9.6%
39.3%
worstSFOC
9.2%
BestSFOC
okSFOC

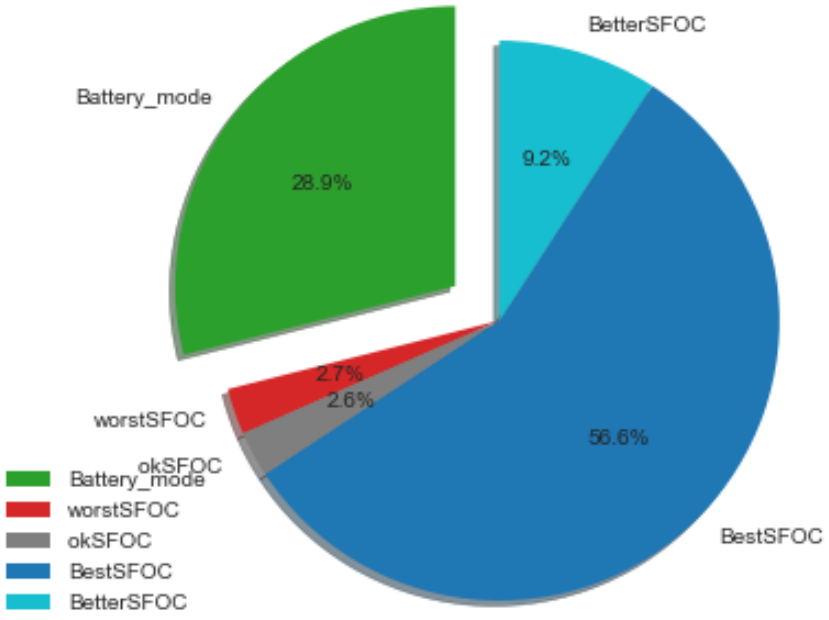
Operation Mode:Training 8000-20000 actions

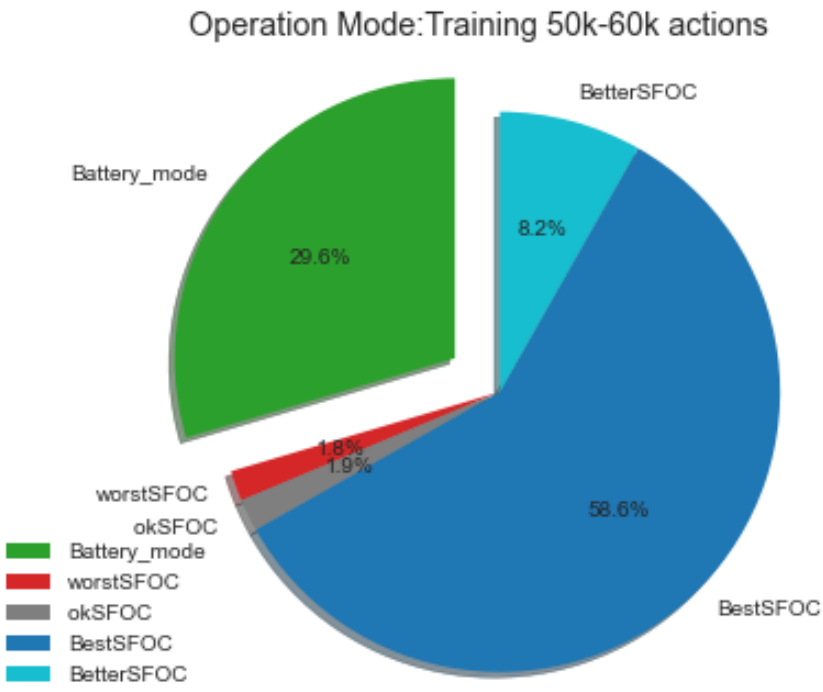


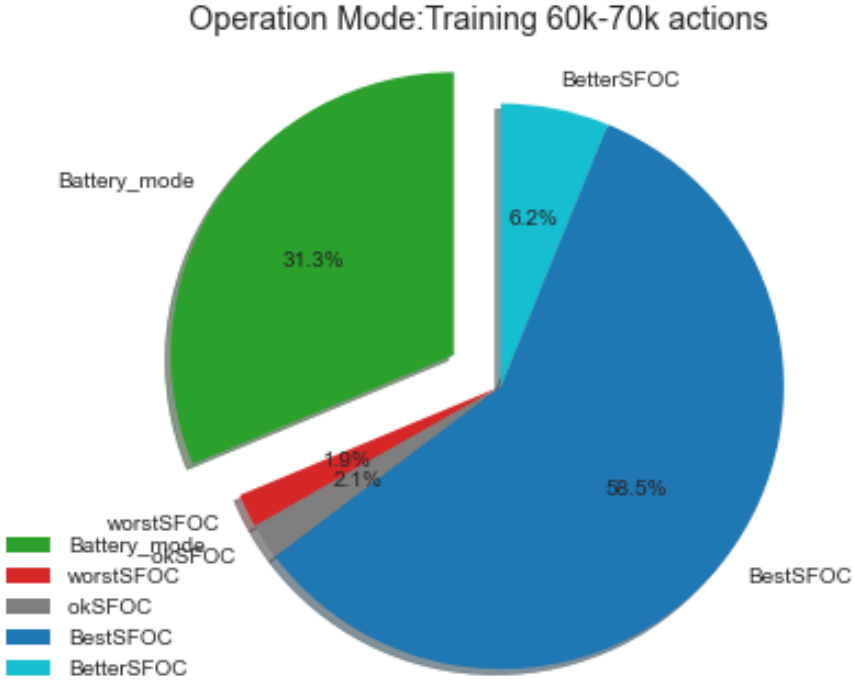
Operation Mode:Training 20000-30000 actions

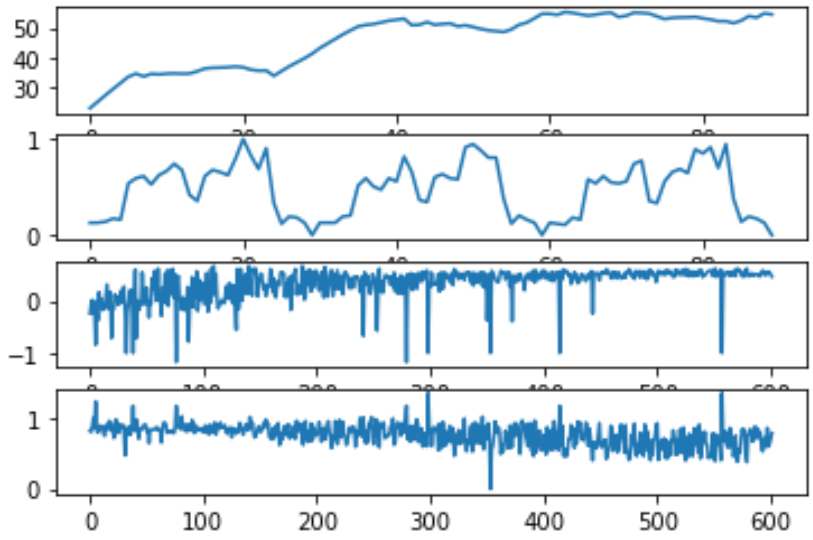



Operation Mode:Training 30k-50k actions
Battery_mode
27.9%
BetterSFOC
13.9%
7.0%
worstSFOC
6.9%
okSFOC
44.2%
BestSFOC
Battery_mode
worstSFOC
okSFOC
BestSFOC
BetterSFOC

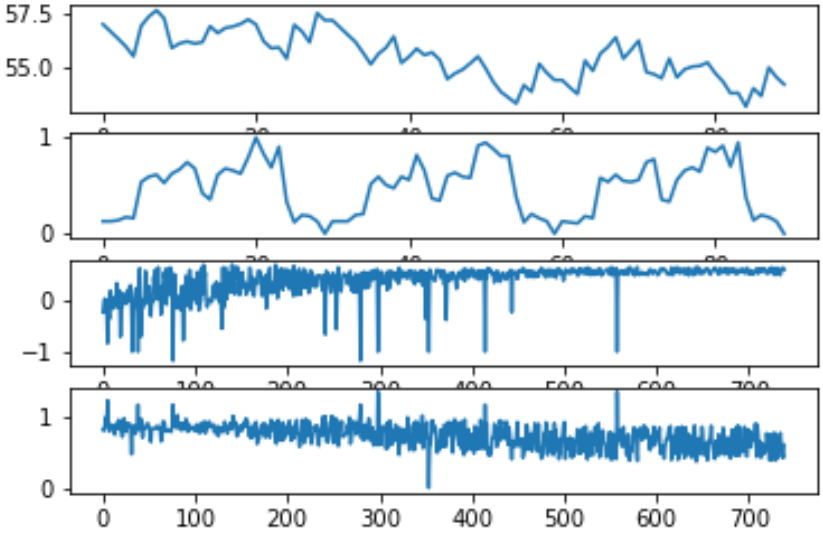

Operation Mode:Training 40K-50K actions
Battery_mode
28.9%
BetterSFOC
9.2%
2.7%
worstSFOC
2.6%
okSFOC
56.6%
BestSFOC
Battery_mode
worstSFOC
okSFOC
BestSFOC
BetterSFOC


Operation Mode:Training 50k-60k actions
BetterSFOC
Battery_mode
8.2%
29.6%
1.8%
1.9%
worstSFOC
okSFOC
58.6%
BestSFOC
Battery_mode
worstSFOC
okSFOC
BestSFOC
BetterSFOC

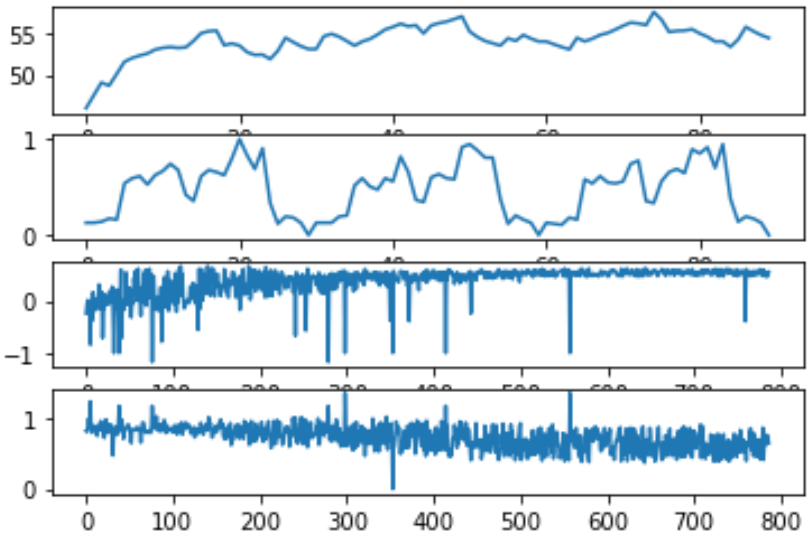

Operation Mode:Training 60k-70k actions
BetterSFOC
Battery_mode
6.2%
31.3%
1.9%
2.1%
58.5%
worstSFOC
Battery_mode
okSFOC
BestSFOC
worstSFOC
okSFOC
BestSFOC
BetterSFOC



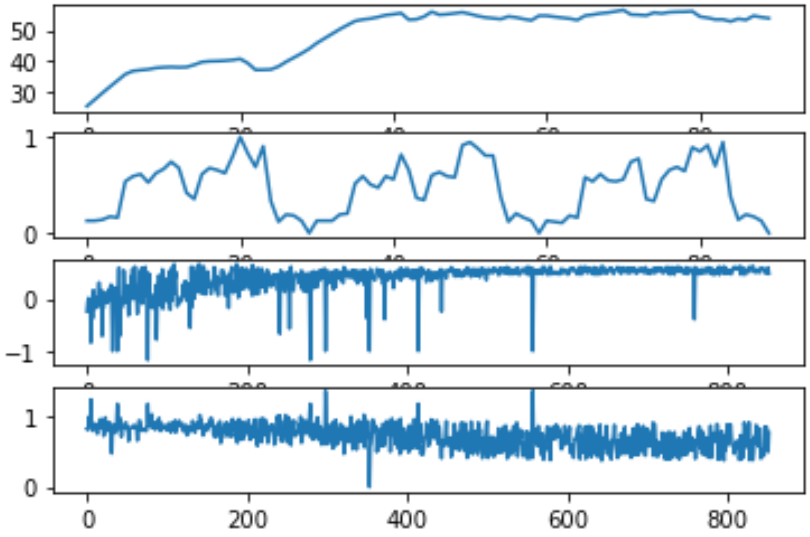
Episode :601



Episode 739



Episode 787



Episode 853

See github repository for full simulation