Neeraj Mehta

Ship trajectory prediction in confined waters.

July 2023

Master's thesis

Master's thesis

2023

Neeraj Mehta

NTNU
Norwegian University of
Science and Technology
Faculty of Engineering
Department of Marine Technology

NTNU
Norwegian University of
Science and Technology

Ship trajectory prediction in confined waters.

Neeraj Mehta

Marine Machinery

Submission date:	July 2023
Supervisor:	Dong Trong Nguyen
Co-supervisor:	Emil Hjelseth Thyri

Norwegian University of Science and Technology
Department of Marine Technology

Neeraj Mehta

Ship trajectory prediction in confined waters

Master’s thesis in Marine Technology
Supervisor: Dong Trong Nguyen
Co-supervisor: Emil Hjelseth Thyri
July 2023

Norwegian University of Science and Technology
Department of Marine Technology (IMT)

NTNU

NTNU Trondheim
Norwegian University of Science and Technology
Department of Marine Technology

MSC THESIS DESCRIPTION SHEET

Name of the candidate:	Neeraj Mehta
Field of study:	Marine Technology (Marine Machinery)
Thesis title (Norwegian):	Forutsigelse av skipsbane i trange farvann.
Thesis title (English):	Ship trajectory prediction in confined waters.

Background

- The motion planning process of a ship is a critical aspect to ensuring safe navigation, particularly in confined waters. The process involves considering multiple factors such as the vessel's draft, depth, distance to land masses, traffic corridors, and potential encounters.
- Relying solely on the current state of the target ship for own motion planning can result in limited decision-making capabilities and a higher likelihood of Low-Level/Reactive COLAV responses. This issue becomes more prominent in confined waters where the risk of collision is significantly increased.
- Despite the significance of motion planning, the existing target ship path prediction algorithms, which are considered the current state-of-the-art, have certain limitations in accommodating non-linear trajectories. This presents an opportunity to develop novel algorithms that can effectively predict the path of target ships in confined waters.
- In summary, the statements above identify the need for improved target ship path prediction algorithms for confined waters and highlights the potential for the development of novel approaches to address this challenge.

Work description

1. Perform a background and literature review to provide information and relevant references on:
    - Existing ship motion prediction algorithms
    - Briefly discuss the results of the literature review conducted during Fall 2022, as a part of the master thesis document.

   Write a list of abbreviations and explain the relevant concepts related to the literature study and project assignment.
2. Develop an effective framework for processing AIS datasets, preparing them for use with machine learning algorithms.
3. Develop a novel method for short-term trajectory prediction based on AIS data as input, accommodating the non-linearities associated with trajectories in confined waters.
4. Train and test the proposed algorithm using unseen AIS dataset.
5. Evaluate the performance of the algorithms in terms of accuracy.
6. Provide an analysis of the results and the limitations of the algorithm and modeling process.
7. Discuss the potential implications of the algorithm for use in real-world navigation and identify areas for future work.

Specifications

The scope of work may prove to be larger than initially anticipated. By the approval from the supervisor, described topics may be deleted or reduced in extent without consequences with regard to grading.

The candidate shall present personal contributions to the resolution of problems within the scope of work. Theories and conclusions should be based on mathematical derivations and logic reasoning, identifying the various steps in the deduction.

The report shall be organized in a logical structure to give a clear exposition of background, results, assessments, and conclusions. The text should be brief and to the point, with a clear language. Rigorous mathematical deductions and illustrating figures are preferred over lengthy textual descriptions. The report shall have font size 11 pts., and it is not expected to be longer than 60-80 A4 pages, from introduction to conclusion, unless otherwise agreed upon. It shall be written in English (preferably US) and contain the following elements: Title page, abstract, acknowledgments, thesis specification, list of symbols and acronyms, table of contents, introduction with objective, background, and scope and delimitations, main body with problem formulations, derivations/developments and results, conclusions with recommendations for further work, references, and optional appendices. All figures, tables, and equations shall be numerated. The original contribution of the candidate and material taken from other sources shall be clearly identified. Work from other sources shall be properly acknowledged using quotations and a Harvard citation style (e.g. *natbib* Latex package). The work is expected to be conducted in an honest and ethical manner, without any sort of plagiarism and misconduct. Such practice is taken very seriously by the university and will have consequences. NTNU can use the results freely in research and teaching by proper referencing, unless otherwise agreed upon.

The thesis shall be submitted with a printed and electronic copy to the main supervisor, with the printed copy signed by the candidate. The final revised version of this thesis description must be included. The report must be submitted according to NTNU procedures. Computer code, pictures, videos, data series, and a PDF version of the report shall be included electronically with all submitted versions.

Start date: 18 February, 2023 **Due date:** 10 July, 2023.

Supervisor:	Dong Trong Nguyen
Co-advisor(s):	Emil Hjelseth Thyri

Trondheim, _________________

____
Dong Trong Nguyen
Supervisor

Abstract

This thesis addresses the challenge of predicting vessel trajectories in confined waters, where the multi-dimensionality of the interaction scene introduces non-linearity in ship paths. The proposed approach utilizes machine learning models trained on processed Automatic Identification System (AIS) data to forecast future vessel trajectories based on observed states.

Until now, latitude, longitude, and other AIS features such as speed over ground (SOG), course over ground (COG) have often been used to train models for target latitude and longitude predictions. However, these models often struggle to capture the inherent non-linearity present in confined water scenarios. To overcome this limitation, the thesis proposes a novel data representation to better capture motion patterns.

The methodology starts with, processing the AIS data using a multi-step framework, maximizing data trueness, retaining valuable information, and minimizing noise. Additionally, learnable features are engineered to improved the mapping of data to target trajectories. Three types of recurrent neural network (RNN) architectures, including Long-Short Term Memory (LSTM), Gated recurrent unit (GRU), and Convolutional Neural Network-Long short term memory (CNN-LSTM), are employed in the study. Two distinct approaches are used for training the models. The first approach uses traditional features such as latitude, longitude, COG, SOG, along with engineered features, mapping them to target latitude longitude coordinates, all in standardized form. The second approach, trains the models to predict relative distance and angle based on observed latitude, longitude and other features. The performance of different approaches and models is evaluated using the accuracy of median target predictions as the key performance indicator.

The results demonstrate that the second approach, which incorporates the novel Relative Displacement and Angle (RDA) data representation, significantly outperforms conventional data representation techniques. Across all models, the RDA approach exhibits approximately 65% improvement in accuracy compared to standard representation methods. The CNN-LSTM model achieves the lowest median deviation error of 0.021 nm for each time-step over a 5 minutes prediction horizon.

Though the RDA approach provides us with a better learnable representation, the models used in this thesis are conventional in nature, with LSTM and GRU being utilized in previous works. Incorporating the RDA approach with more sophisticated models has the potential to further enhance accuracy. Additionally, the thesis acknowledges the limitation of not considering encounter data, which can impact the accurate estimation of vessel intent/trajectories in encounter scenarios.

Preface

This thesis marks the culmination of my M.Sc degree in Marine Technology (Marine Machinery) at the Norwegian University of Science and Technology (NTNU). It has been an enriching journey, giving me an opportunity to explore my interests and learn about the fascinating intersection between the fields of machine learning and maritime . Despite encountering challenges along the way, I can confidently state that the ongoing advancements in maritime autonomy have been a great source of motivation, and I feel privileged to having the opportunity to contribute in the domain.

I would like to express my heartfelt gratitude to my supervisor, Dong Trong Nguyen, and my co-supervisor, Emil H. Thyri, for their invaluable guidance and support throughout this endeavor. Despite my background in marine machinery, they provided me with the opportunity to choose this topic and helped shape the development of this work. Our regular biweekly meeting during the semester proved instrumental in discussing ideas, addressing concerns and making progress.

I would also like to extend my appreciation to Stian Molden and Bjørn Tore Bach for their assistance with the datasets and computational resources. Their contributions have been instrumental in carrying out the necessary experiments and analyses for this work. I am also grateful to the faculty and staff at NTNU for creating a conducive learning environment and providing the necessary resources for my academic growth. The open learning environment has played a key role in discovering my interests and fostering my passion in the space.

In addition, I would like to express my deepest gratitude to my family and friends, for their unwavering belief in my abilities and their continuous support throughout my studies. Your encouragement and understanding have been an invaluable source of motivation, helping me overcome challenges along the way.

Reading through the literature on existing works, I am grateful to all the researchers, and industry professionals who have contributed to the advancement of machine learning and its applicability in the maritime domain. Their collective actions have paved the way for exciting innovations in the field. It is my hope that the findings presented in this thesis will contribute to existing body of knowledge and inspire further progress in the domain of trajectory prediction, particularly in confined waters.

Moreover, this work is done in collaboration with Zeabuz, with Emil H. Thyri as my co-supervisor. The limitations identified in Thyri's previous work as a part of PhD thesis, have been key to identifying the objective of this thesis.

Table of Contents

List of Tables

List of Figures

Abbreviations

Abbreviation	Explanation
AIS	Automatic Identification System
COLREGs	International Regulations for Preventing Collisions at Sea 1972
COLAV	Collision Avoidance
CPA	Closest Point of Approach
CNN	Convolutional Neural Network
EKF	Extended Kalman Filter
GMM	Gaussian Mixture Model
GRU	Gated Recurrent Unit
LSTM	Long Short-Term Memory
nm	Nautical Mile
RDA	Relative Displacement Angle
RNN	Recurrent Neural Network
SOG	Speed Over Ground
SVR	Support Vector Regression
TCPA	Time to Closest Point of Approach
TSS	Traffic Separation Scheme
MSE	Mean Square Error
CE	Cross Entropy
Adam	Adaptive Moment Estimation
COG	Course over ground
WGS84	World Geodetic System 1984
NDC	Nearest Distance to Coast
ML	Machine Learning
CNN-LSTM	Convolutional LSTM
MLP	Multi-layer perceptron

Chapter 1

Introduction

This chapter starts with the motivation behind undertaking this thesis. Subsequently, an overview of existing literature with their theories and findings is discussed. Furthermore, a summary of the contributions made by this thesis is presented, emphasising the significance of the approach. Finally an outline is presented to give a preview of the subsequent sections.

1.1 Motivation

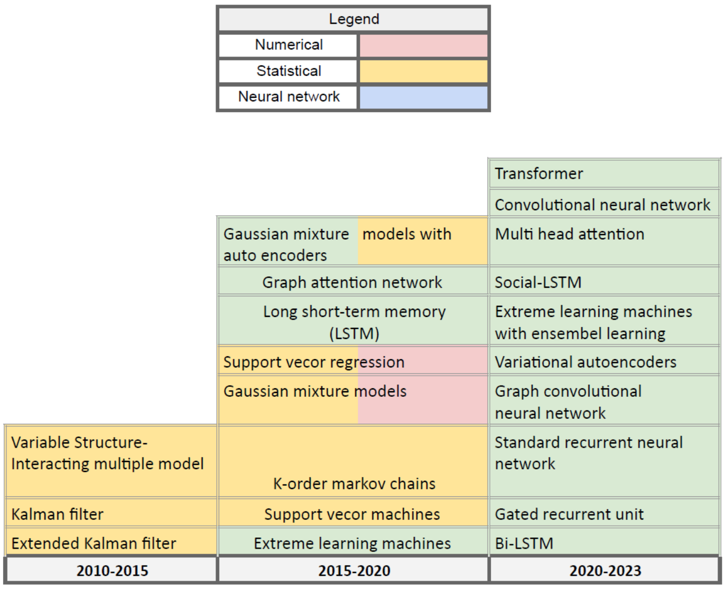
The motivation for this thesis stems from the works of Thyri et al. (2020) and the recent technological advancements that are facilitating the emergence of a new era characterized by automation. This automation in many scenarios, demands the integration of existing sensors with advanced data analytics, rather than a sole reliance on advanced censor technologies. As we strive towards achieving the objectives of industry 4.0, our focus extends beyond the scope for increased productivity and profitability, to enhancing the human and environmental safety, Rosin et al. (2022).

For the maritime transport sector this increased safety can be achieved with systems that enable the implementation of safety-critical decisions in real-time. Among thesis decisions navigation plays a crucial role. This is evident in a report issued by EMSA (European Maritime Safety Agency), where it was fount that "More than half of the casualties with a ship (54.2%) were related to issues of a navigational nature, such as contacts, grounding/ stranding, and collisions ", European Maritime Safety Agency. Furthermore, 65.8% of the total 4104 reported maritime accidents for 2011-2018 were attributed to human action. Similarly on land, road crashes account for about 1.3 million deaths and 20-50 million non-fatal injuries each year, NHTSA. The adoption of autonomy in land navigation has gained widespread acceptance as it aims to improve the safety on roads. Implementation of a similar model in the maritime domain offers several advantages. The European Environmental Agency, in one of its publications stated out that “rail and waterborne transport are much more greenhouse gas efficient than road transport and aviation, both for passengers and freight" , European Environmental Agency. Thus, the adoption of maritime autonomy around urban areas can help decongest cities, improve human and environmental safety, increase efficiency, provide round-the-clock accessibility, and generate cost savings in terms of both manning and operation.

In order to implement safe navigational autonomy in the maritime space, it is crucial to consider the cognitive aspect of human decision-making. This consideration becomes particularly important in confined waters, where the multi-dimensionality significantly increases. This dimensionality is bought in by factors such as distance to the coastline, depth along the intended course, interactions with multiple agents, and other relevant scenes. More recently, many trajectory and path prediction models are being developed, with many proving efficient for open sea operations, where linear movement patterns are prevalent, as opposed to coastal/confined waters.

1.2 Literature Review

With the advent of GPU and parallel computational technologies advanced machine learning techniques are being developed around data driven models. A similar trend can be seen in the maritime space, and also in the context of trajectory prediction. Figure 1.1 illustrates that, in recent times deep neural networks have been gaining greater acceptance compared to traditional statistical and numerical models.



Legend	
Numerical	🟥
Statistical	🟨
Neural network	🟦

		Transformer
		Convolutional neural network
	Gaussian mixture models with auto encoders	Multi head attention
	Graph attention network	Social-LSTM
	Long short-term memory (LSTM)	Extreme learning machines with ensembel learning
	Support vecor regression	Variational autoencoders
	Gaussian mixture models	Graph convolutional neural network
Variable Structure-Interacting multiple model	K-order markov chains	Standard recurrent neural network
Kalman filter	Support vecor machines	Gated recurrent unit
Extended Kalman filter	Extreme learning machines	Bi-LSTM
2010-2015	**2015-2020**	**2020-2023**

Figure 1.1: Timeline- Trajectory prediction algorithms

As highlighted in previous section 1.1, the multi-dimensionality significantly increases around coastal regions, leading to a notable impact on vessel behaviour. Consequently this leads to a large part of trajectories around coastal regions being non-linear. While large vessels typically adhere to traffic separation scheme (TSS), following certain combinations of speed and heading, smaller vessels have lesser privileges with-respect-to the right of way and have to be more flexible in path planning, accounting for traffic and ensuring collision avoidance. In a statistical model approach, Perera and Guedes Soares (2010) use Extended Kalman filter (EKF), capturing the non-linearity in trajectory motions which the conventional kalman filter cannot. The non-linear continuous-time target motion model and the discrete-time linear measurement model are initialized with simulated trajectory states, followed by iterative projection and updates to state estimates, errors and, gain. Though in the stated work the model is able to effectively capture non-linearity and gain convergence to velocity and acceleration states, accuracy reliability can still be a challenge. This is attributed to, heavy reliance on the initial state estimate for convergence, the requirement for accurate modeling of complex equations to represent motion models, and the assumption that the target motion and measurement models follow a gaussian distribution. In another statistical approach, Krishanth et al. (2012) present work addressing a particular use cases involving transit within shipping lanes. The proposed Variable Structure- Interacting Multiple Model (VS-IMM) combines multiple models, each representing a different motion behaviour. The models are switched dynamically based on the target's movement across different segments. In cases involving multiple junctions or complex scenarios, the authors suggest utilizing a segment hypothesis tree to retrodict and trace back the target's location. However, the algorithm's effectiveness is limited to junction segment probability estimation and not intended for non-linear trajectory prediction. "SVR seaway", a model proposed by Joo-Sung Kim (2017), is statistical in nature and suited towards a particular use case. The support vector regression (SVR), an extension of the commonly used support vector machines for classification, is employed in this model. SVR maps input features onto a kernel function and seeks to identify the optimal hyperplane by maximizing the margin around support vectors. The author uses position, speed and course data to train the model and extract non-linear hyper-planes for both the Closest Point of Approach (CPA) and Time to Closest point of Approach (TCPA) predictions. The reliance on the need to store support vectors, makes the model memory intensive and impractical for large datasets. With a similar algorithm, Liu et al. (2019) uses the SVR with adaptive chaos differential equation optimizer for faster convergence. The SVR for this case shows good convergence over next single step prediction using the current and the three previous time-steps. However, as before the SVR's dependence on support vectors makes it sensitive to changes in the distribution of vector positions, and may introduce significant errors when encountering previously unseen scenarios, including multi-step predictions.

Another, single step prediction model is proposed by Guo et al. (2018), however specifically for ocean spaces. The algorithm consists of a k-order multivariate markov chain, modeling the trajectory prediction task as an event-outcome model. The 3 step process for prediction involves, first dividing the sea area into non-overlapping grids, then choosing vessel location, direction and speed as key seen states, and finally calculating the transition probability matrix and, predicting the next probable grid cell representing the unseen position state. From analysis results and a comparison between the 1st, 2nd and 3rd order markov chain, as the chain length increases ie. number of past seen events, the prediction accuracy improves, giving the highest for 3-order model. However, it shall be noted that while capturing more past states enhances memory, it also significantly increases the computational complexity, particularly for time-dependent probabilities. Even with the growing available compute, the markov chains heavily reliance on transition probabilities and current states, makes them weak at discovering complex long term dependencies as well as performance on undiscovered and dynamic coastal data.Mao et al. (2018) propose a fast learning neural network algorithm, Extreme learning machines. It consists of a simple 3 layers architecture - input, hidden and output layers, and use a single pass learning approach. The algorithm analytically calculates the output weights using simple linear regression, which is based on randomly generated weights in the hidden layer. This simplicity in both, the architecture and learning process, contribute to the model's quick training time and satisfactory generalization performance. However, these attributes introduce drawbacks such as, absence of iterative fine-tuning and, the inability to capture complex feature and pattern correlations.

In a probabilistic approach to trajectory prediction, Gaussian mixture models (GMM) is proposed by Dalsnes et al. (2018). First, the neighbour course distribution method is used to construct a tree of states, which represents multiple trajectories. Subsequently an expectation maximization algorithm is employed to fit a GMM to these points, resulting in a probabilistic model that predicts the future positions. This model offers several advantageous features, including the ability to associate uncertainty with predictions. However, GMM are inefficient in capturing feature correlations that do not exhibit a normal distribution. This can be to disadvantage in coastal areas where the trajectories are often skewed, as seen for our data represented in figures 1.2 and 1.3. Moreover, the results by the authors indicate that the model suffers in areas with sparse data density, leading to overconfident yet inaccurate predictions. This drawback can be attributed to the model's heavy reliance on mapped covariance matrix.

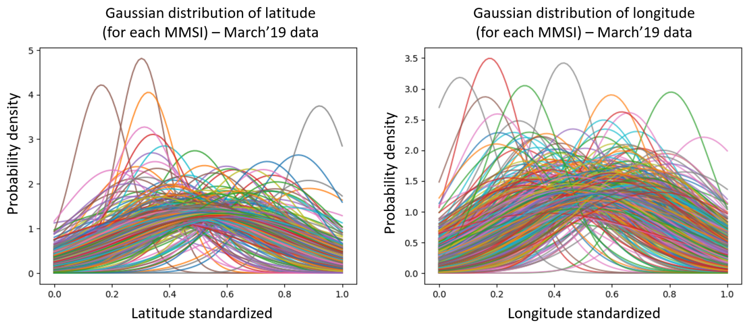


Figure 1.2: Gaussian distributions of standardized position coordinates a)Latitude b)Longitude (Processed march'19 AIS data)

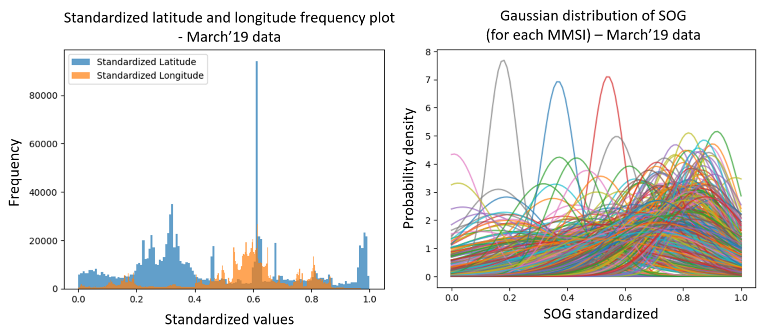


Figure 1.3: a)Standardized geo-coordinates frequency plot b)Gaussian distribution of standardized SOG (Processed march'19 AIS data)

In recent years neural networks have gained popular attention, with some of these architectures being applied to the problem of sequential trajectory prediction. Tang et al. (2019) use a 2 layer LSTM architecture model to observe 10 steps spanning a duration of 10 minutes, and output a single step prediction of position after 10 minutes. From author's results the LSTM model outperformed both the EKF and the back propagation neural networks in terms of prediction accuracy. This can be attributed to the LSTM's ability to effectively capture long-term dependencies in sequential data, making it an attractive choice for solving sequential problems. Murray and Perera (2020) also use a neural network model, dual linear auto-encoder. The authors use a gaussian mixture model to cluster trajectories and assigned the relevant vessels to specific clusters. Then the state information of the vessel is fed to the trained auto-encoders corresponding to that cluster. With this the model is capable of generating possible future trajectories. Due to the compression and latent space projection, the encoders can effectively cut down on noise, helping the decoders extract the important features and their inter-dependencies. Though, the linearity in the proposed model makes it computationally efficient, it may fail at capturing complex and non-linear data. On similar principles, but with an RNN approach, Capobianco et al. (2021) propose a LSTM encoder-decoder architecture. They trained their model on both unlabeled and labeled (with destination intent) trajectory sets.A performance comparison is made against Linear and Multi-layer perceptron models, with Encoder-decoder model performing the best. Taking on a recurrent neural network approach, Suo et al. (2020) use a GRU based framework. A Density-Based Spatial Clustering of Applications with Noise algorithm is used to identify high density trajectories. These identified trajectories are then used to optimize the coordinates of the incoming dataset, using symmetric segment path distance. This optimized set of coordinates is fed to the GRU model, which recursively generates the next step predictions for the trajectories. From results, the authors conclude that the GRU gives accuracy near similar and in some cases slightly better than the LSTM, and significantly better than EKF. However it is worth noting that the evaluation solely relies on the mean squared error, which is an incomplete representation of distance errors. A more practical measure of a trajectory model's performance could be average displacement error (ADE), which have commonly been used for land based scenariosAlahi et al. (2016). Also, the author uses similarity measurement as a part of valid data selection process, which could make the data too refined and though improve accuracy, can lead to loss of generalization and limited adaptability over real-world trajectories. Murray and Perera (2021), in another one of their works, use a deep learning framework based on RNNs. First, Hierarchical Density-Based Spatial Clustering of Applications with Noise is used towards generating trajectory clusters. Then, the trajectory steps are passed through a GRU based Variational auto-encoder and a soft-max classifier, outputting a set of probable trajectories, ie. assigning it one of the cluster classes. It is noted that, representing global ship behaviour with multiple local models trained on multiple small clusters is able to better capture step features, giving better performance compared to global models with and without attention. However, the architecture is sensitive to input number of clusters, making it difficult to capture complex behaviours and patterns. Volkova et al. propose a artificial neural network model for a specific use case, reconstructing missing AIS coordinate trajectory. Two of the key limitations that authors note are, the possibility to make trajectory predictions only one coordinate ahead and, inability to predict non-linear cases.

LSTM architecture has been a popular choice in sequence modelling, and been used in recent trajectory prediction works as well. Some of these using the framework include, Genetic Algorithm optimized LSTM by Qian et al. (2022), encoder-decoder model by Capobianco et al. (2022), SFM-LSTM by Liu et al. (2022). The model by Capobianco et al. is able to take uncertainty into account using Monte Carlo dropout, improving the real world representation as open waters offer multiple possible maneuvers and trajectories. However, the model is executed over a long term horizon and a with a sampling frequency of 15 minutes, which gives a global path prediction and would be less relevant for local and encounter scenarios. In works by Liu et al., each vessel is modeled as a LSTM block, training them over a collision-free social pooling layer, thus having encounters considered towards predictions, though its performance in such scenarios has not been measured. The authors note that the changes in sailing course, introduce difficulty in generating satisfactory prediction. From the comparison made, the model is shown to perform better than clustering and RNN only models. Taking a classification approach, Syed and Ahmed (2023) use a CNN-LSTM model for marine vessel track association, by having each mmsi as a unique vessel id and then training the model to predict the the unique vessel id based on the trajectory inputs. The CNN with use of convolutional layer is able to extract local features and thus capture spatial co-relations, while the LSTM helps capture the long-term dependencies of these key features. Though a classification model itself has limited applicability in ship trajectory prediction, replacing the softmax with a linear/ non-linear activation functions can make the CNN-LSTM relevant for the use case of regression modelling.

Attention based models which have proven to be the state-of-the-art in natural language processing and computer vision applications, are also looked at in recent works. Bao et al. (2022) propose a Multi-head attention-BiGRU model, with take advantage of the computationally efficient GRU combined with attention mechanism to attend to part of sequence and learn more complex dependencies. A long term trajectory prediction model by Nguyen and Fablet (2023) uses a Decoder only transformer architecture, which takes advantage of self-attention mechanism to attend to part of input sequences and weigh their significance when making predictions. Also, the authors use voyages with a minimum duration of 4 hours as a part of the dataset and re-frame the prediction as a classification problem. Though the model uses a recursive step prediction method and thus experiences error propagation with steps, the authors note that the model is able to perform efficiently, with an error of less than 10 nautical miles for a 10 hour duration. As, the proposed model uses a recursive prediction method, the error is shown to increase over time. It shall be noted, that long term trajectories often involve global routes while short term trajectories have a higher dependence on the immediate states and interactions. Also, handling of numerical values and state vectors can be challenging for transformers, as future state vectors (latitude, longitude, COG, SOG combinations) can often be unmapped in available datasets, making it difficulty to map and train over the complete possible set of vocabulary.

1.3 Objective and Scope

The purpose of this work is to investigate, develop, and evaluate algorithms for trajectory prediction in confined waters. Existing studies have predominantly focused on implementing various architectures, often derived from sequence modeling techniques employed in other application domains, many with origins in natural language and image processing. While this evolution process of architectures has led to improved prediction accuracies and enhanced suitability for confined scenarios, less attention has been given to the characteristic representation of the dataset used and its impact on model training.

The objective of this thesis is to propose a method to predict the trajectory of a ship. Based on this objective, the following research questions will be addressed:

1. How can AIS datasets be effectively processed to extract relevant information for the ship trajectory prediction?
2. Is it possible to engineer features from AIS data in conjunction with open available resources?
3. Can machine learning methods predict the ship motion based on available AIS data, and how do they compare in terms of performance?
4. How can trajectory predictions account for observed changes in course for vessels navigating in confined waters?
5. What is an effective dataset representation that can better capture confined water trajectories?

The scope of the work are as follows:

- Perform a literature review of existing trajectory prediction algorithms, and present their advantages and limitations.
- Pre-process the raw AIS dataset to ensure its suitability for machine learning algorithms.
- Develop and implement machine learning algorithms capable of learning long term dependencies in ship trajectories.

- Model the system for application of trajectory prediction in confined waters, considering the non-linearity in motion patterns.
- Configure the developed model to make short-term predictions based on a time window, where the duration of seen states is 5 minutes and the futre is is 5 minutes as well.
- Identify potential areas for improvement and future work.

By working on these objectives and scope, the work aims to contribute to the improved performance of trajectory prediction methods in confined waters, facilitating safer and more efficient navigation.

1.4 Contributions

Some of the contributions of this thesis include:

- Multi-step Data Processing Framework: A data preprocessing framework is developed, which incorporates a multi-step approach to maximize the retention of useful information while filtering out noise and irrelevant data. This framework ensures that the input data for trajectory prediction models is properly prepared, leading to improved accuracy and reliability of the models.
- Novel RDA Approach: This thesis introduces a novel Relative Displacement Angle (RDA) approach for data representation in short-term trajectory prediction models. The RDA approach reduces vocabulary and improves the mapping of relative angle information, enabling better learning of state inter-dependencies and capturing non-linear trajectory patterns more effectively.Novel RDA Approach: This thesis introduces a novel Relative Displacement Angle (RDA) approach for data representation in short-term trajectory prediction models. The RDA approach reduces the vocabulary and enhances the mapping of relative angle information, enabling the models to learn state inter-dependencies more effectively and capture non-linear trajectory patterns more accurately. The RDA approach significantly improves the performance of the prediction models.
- Insights into Model Selection and Parameter Tuning: Through the evaluation and comparison of multiple models, this thesis provides insights into the process of model selection and parameter tuning for short-term trajectory prediction in confined waters. By analyzing the architectures, performance metrics, and learning capabilities of different models, informed decision can be made regarding the kind of model and optimal parameter settings.
- Evaluation and Comparison of RNN Models: The thesis conducts a comprehensive evaluation and comparison of three popular Recurrent Neural Network (RNN) models - LSTM, GRU, and CNN-LSTM - for short-term trajectory prediction in confined waters.

- Identification of Future Research Directions: The thesis identifies several promising research directions to advance the field of trajectory prediction in confined waters. These include the consideration of encounter scenes, exploration of advanced modelling approaches, incorporation of physics-based models such as spline functions, and enhancement of feature engineering techniques. Lastly, the thesis highlights the need for the development of an open-source trajectory dataset specific to confined waters, which can serve as a standardized benchmark for future research in this area.

1.5 Outline

The rest of the thesis is structures as follows: Chapter 2 provides a comprehensive review of the theoretical foundations underlying machine learning models and hyperparameter tuning techniques. Chapter 3, presents the exploration and preprocessing of the data. The methodology employed for the RDA approach and the implementation of machine learning models are detailed in Chapter 4. Chapter 5 presents the results obtained from our research. The discussions are further presented in Chapter 6. The concluding remarks and avenues for future research are presented in the final chapter.

Chapter 2

Theory

This chapter begins by identifying the problem type associated with ship path prediction in confined waters. An overview of existing trajectory prediction algorithms in the maritime domain is then provided, along with a discussion on their types and the objective properties used to assess their effectiveness. Subsequently, the theoretical foundations of the models selected for this thesis are examined.

2.1 Trajectory prediction problem

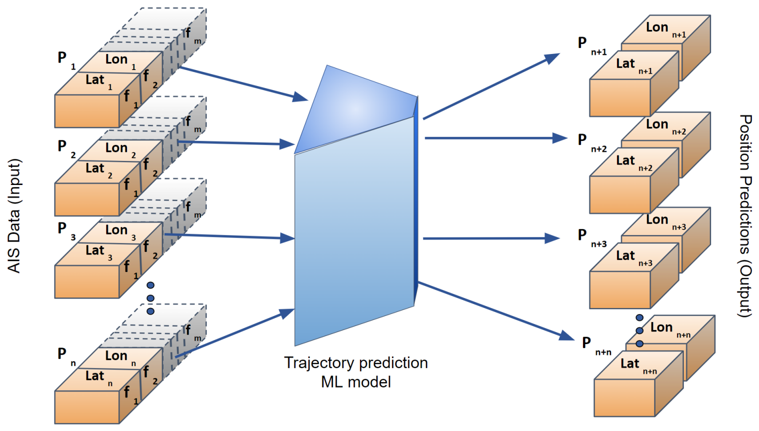


Figure 2.1: Trajectory prediction model-input and target state representation

The first step to using a ML model, is to frame the ML problem. As our scope of study does not involve evaluating the impact of such a trajectory prediction model in the real world, thus we use a modified version of "Framing a ML problem" , Google. The 3 tasks involve:

- Defining the outcome and the model's goal- As shown in figure 2.1, the trajectory prediction model's goal shall be to take in n number of previous time-step AIS data and predict the next n time-step positions.

  In this case, n takes the value of 10 steps with the time-interval between each being 30 seconds. An ideal outcome, would be minimal deviation between the predicted and the true future positions of the vessel.

- Identify the model's output- A model's output shall be aimed towards achieving the ideal outcome. In this case it is aimed at case making accurate multi-step sequence predictions, i.e. future position co-ordinates - latitudes and longitudes. These predictions would be of type, floating point values. Additionally, as shown in Figure 2.1, multiple AIS features are considered in the prediction process. Therefore, a multivariate multi-step regression model is deemed the most suitable choice. Considering the above two requirements a multivariate multi-step regression model would be the most appropriate choice.

- Define success metrics- To measure the model's performance and conduct a comparative analysis, three metrics are adopted to quantify the deviation between the predicted and actual future trajectory coordinates. The employed metrics are as follows:

  – Average/Mean displacement error: As shown in equations 2.1 , it gives a mean of the distance error between the predicted and true trajectory coordinates. A lower value of average displacement error demonstrates a higher level of prediction accuracy, however the same may be sensitive to outliers.

  $$ADE = \frac{1}{T}\sum_{i=1}^{T}\sqrt{\left(x_i^{Pred.} - x_i^{GT}\right)^2 + \left(y_i^{Pred.} - y_i^{GT}\right)^2} \tag{2.1}$$

  where,
  T constitutes total number of trajectory prediction indices
  with, ($x_i^{Pred.}$, $y_i^{Pred.}$) as the predicted (latitude, longitude) values and
  (x_i^{GT}, y_i^{GT}) being the ground truth values for the corresponding index.

  – Median displacement error: It measures the median distance between the predicted and true trajectory coordinates. Similar to clustering, taking advantage of populating values, the metrics is less susceptible to outliers and gives a better general estimate of prediction errors.

  – Final displacement error: It measures the offset distance between the final prediction step and the ground truth coordinates. As discussed in section in a recursive step prediction method the error can often propagate over time-steps leading to final displacement error as the largest offset distance from ground truth, relative to previous prediction steps.

2.2 RNN based neural network models

A neural network in its basic form, replicates the information flow architecture in a human brain, consisting of neurons and their inter-connections. The neurons acts as computational blocks that perform a function over given input(s) and generate an output, as shown in figure 2.2. The output then flows thought weighted connections, inputting information to another neuron in the network. The weight associated to a connection determines the importance of information that flows thought it.

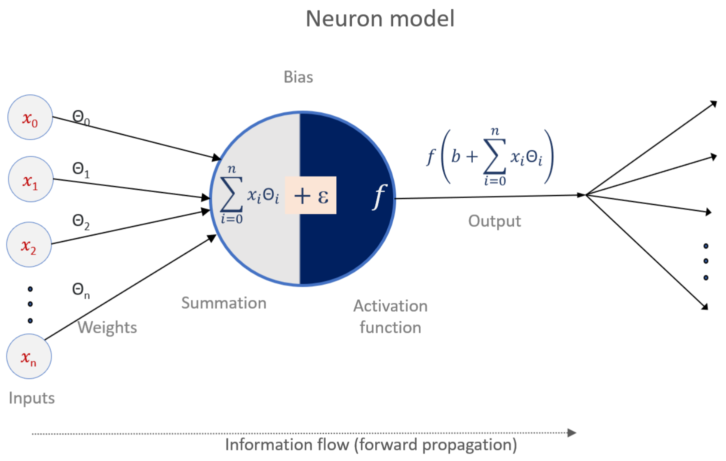


Figure 2.2: Feed forward neuron architecture

Stacking these neurons together and having multiple layer of inter-connected neurons makes the multi-layer perceptron (MLP), which is capable of learning more complex representations and features in the training data. A simple MLP learning procedure involves a 3 step process, Mayank Banoula:

1. Forward propagation: Pass the data along the information flow channel, and across layers, finally to the output layer.

2. Gradient calculation: The predicted output is then compared against the ground truth, to calculate the error.

3. Back propagation: The error is then back-propagated to update the weights of the channels. This weight optimization is carried out using algorithms such as gradient descent, stochastic gradient descent, adaptive learning rate methods etc. A gradient descent update step can be given by equation 2.2:

$$\Delta_{\Phi}(t) = -\epsilon \frac{\partial E}{\partial \Phi_{(t)}} + \alpha \Delta_{w(t-1)} \tag{2.2}$$

where, $\Delta_{\Phi}(t)$ is the current gradient, ϵ is the bias, $\frac{\partial E}{\partial \Phi_{(t)}}$ is the change in error with respect to change in weight vector, α is the learning rate, and $\Delta_{w(t-1)}$ is the previous iteration gradient.

These step are then iterated over, to optimize the weights towards minimizing the cost function/error. While MLPs might perform well on some classification tasks involving sentiment analysis, image classification, and regression such as ship performance/ water resistance prediction, they however are constrained to fixed-size inputs and with input vectors having temporal independence. Thus, for our spatiotemporal problem it has limited applicability.

2.2.1 Standard RNN

An important development that came up later in the field of NN was the RNN, introducing the concept of memory. This concept enables the learning of sequential dependencies in data. Compared to traditional NN, standard RNNs use a looping mechanism as shown in figure 2.3, transferring information from previous steps to the current step, helping the model capture temporal relationships.

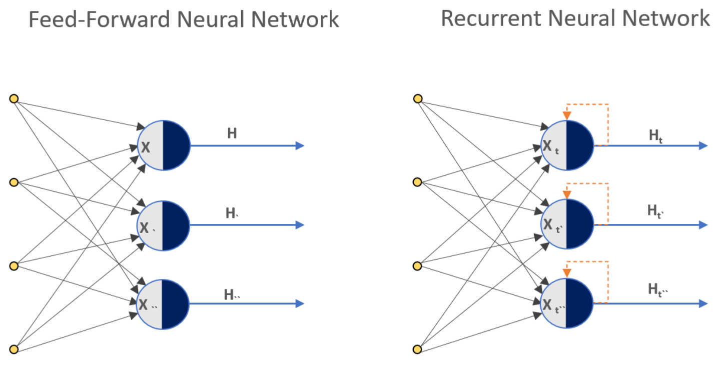


Figure 2.3: Feed-forward NN vs RNN

This recurrence however also means that gradients are propagated back also in time, causing exponential decline of gradient magnitude over previous time-step due to weight multiplication. This makes the RNN suffer from vanishing gradient problem, limiting the learning of long-term dependencies, thus making it unsuitable for cases where the context lies not near the current time-step but extends over significant temporal span.

2.2.2 LSTM

Overcoming the limitations of standard RNN are a special kind of RNN, LSTM introduced by Hochreiter and Schmidhuber (1997). Shown in figure 2.4, two key innovation in the LSTM are the memory cell and the cell state channel architectures, helping the model preserve long-term dependencies by allowing the flow of gradients without significant decay.

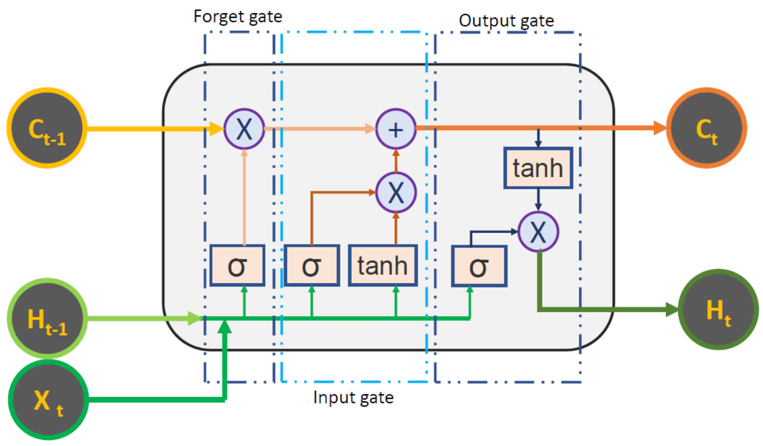


Figure 2.4: LSTM cell, adapted from Christopher Olah

The flow of information is controlled by gate structures in the cell. Each of the 3 gates consist of a sigmoid and point multiplication operators.The sigmoid outputs values between 0 and 1, with 0 indicating complete non-relevance of information and 1 indicating some degree of relevance, with respect to the previous cell state. The point multiplication operators control the weights and thus the importance of the input information. This combined operation ensures information flow control by scaling gate inputs as per relevance.

The three gates in this architecture are:

- Forget gate- As the relevance of states close to current moment increases, it becomes important to forget the information from distant temporal span, to ensure better capture of dependencies that are more relevant and closer in time. This is executed using the forget gate, selectively suppressing the information from the previous cell state by multiplying it with output from the sigmoid function, shown in equation 2.3, Greff et al. (2017):

$$f_t = \sigma(W_f * X_t + R_f * H_{t-1} + b_f) \tag{2.3}$$

  where, f_t is the activation output of forget gate, W_f and R_f the input and recurrent weights respectively, X_t the current input, H_{t-1} the previous hidden state, and b_f is the forget gate bias.

- Input gate- Next step involves calculating the new information for updating the cell state. A part of the step is given by equations 2.4 and 2.5, which involves using the sigmoid (σ) function and hyperbolic tangent ($\tanh$) activation block to help calculate the relevance of current input and previous output. The new relevant information, given by equation 2.6 is used to update the cell state.

$$i_t = \sigma(W_i * X_t + R_i * H_{t-1} + b_i) \tag{2.4}$$

$$N_t = \tanh(W_c * X_t + R_c * H_{t-1} + b_c) \tag{2.5}$$

$$I_t = i_t \odot N_t \tag{2.6}$$

Shown in equation 2.7, the cell state is now updated with the new information.

$$C_t = f_t \odot C_{t-1} + I_t \tag{2.7}$$

Where, C_t is the cell state after update, f_t the forget value, C_{t-1} the previous cell state, and I_t is the new information value.

- Output gate- This gate executes the final step in an LSTM cell, determining the part of information from the updated cell state that shall be outputted. Similar to input gate, it consists of the sigmoid (σ) and the hyperbolic tangent ($\tanh$) activation functions. The filtered output from the σ block is applied over the derived cell state from the $\tanh$ activation function. Equations 2.8 and 2.9 represent this output generation process.

$$o_t = \sigma(W_o * X_t + R_o * H_{t-1} + b_o) \tag{2.8}$$

$$H_t = o_t \odot \tanh(C_t) \tag{2.9}$$

where, o_t and H_t represent the outputs from the gate's sigmoid block and the LSTM cell respectively. W_o and R_o are respectively the weights of the input and recurrent connections of output gate, and b_o is the output gate bias.

2.2.3 GRU

Another variant of specialized and learnable RNN is the GRU, introduced by Cho et al. (2014).

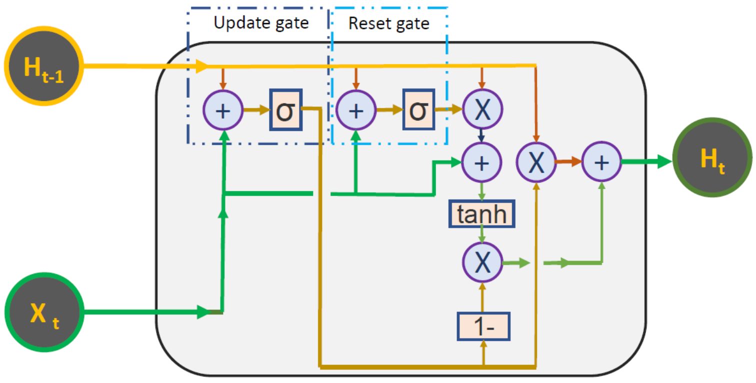


Figure 2.5: GRU cell, adapted from Daoulas et al. (2021)

Similar to LSTM, it uses a gated architecture which is more simpler while also giving comparable performance. As shown in figure 2.5, the GRU cell consists of 2 gates:

- Update gate- Similar to the input gate in the LSTM block, the update gate helps the model to measure the relevance of information from the previous cell state and the current input, and accordingly pass it to the future. This helps the model prevent vanishing gradient problem, by enabling it to selectively retain the information from previous hidden states. The update step can be given by equation 2.10:

$$z_t = \sigma(W_z * X_t + R_z * H_{t-1} + b_z) \tag{2.10}$$

  where, r_t represents the update gate, X_t and H_{t-1} are the current input and previous hidden state respectively with, W_r and R_r being their corresponding weights. b_r is the bias term for the reset gate.

- Reset gate- Similar to forget gate in the LSTM block, the reset gate in GRU is used to determine the extent of diminishing the influence of past output. Thought the formula as shown in equation 2.11 looks similar to the GRU-update gate, the difference in gate arises from the difference in weights and thus their outputs, and ultimately their effects within the GRU architecture.

$$r_t = \sigma(W_r * X_t + R_r * H_{t-1} + b_r) \tag{2.11}$$

  where, r_t represents the reset gate, W_r and R_r being weights applied to input and previous hidden states for the particular section. Further, b_r is the bias term for the reset gate.

The sigmoid (σ) functions from both the gates size the information and output values between 0 and 1. As shown by equation 2.12, the reset gate output is multiplied element-wise with the previous hidden state, to filter out non-relevant information from the past time steps. The remaining useful past information is then added to the current input, and a hyperbolic tangent ($\tanh$) activation function applied over it to capture the information in a new hidden state ($h_t^{'}$).

$$h_t^{'} = \tanh(W_{h^{'}} * X_t + r_t \odot R_{h^{'}} * H_{t-1}) \tag{2.12}$$

To generate the final hidden state of the cell (H_t), the update gate output is used to select the relevant information from both, the new hidden state ($h_t^{'}$) and the hidden state from previous time-steps (H_{t-1}). Equation 2.13 demonstrates this hidden state calculation.

$$H_t = z_t \odot H_{t-1} + (1 - z_t) \odot h_t^{'} \tag{2.13}$$

2.2.4 CNN

The key foundation behind CNNs can be found in a paper by Fukushima (1980), with the proposal of a neocognitron model which mimics the brain process when subject to visual information, making use of receptive field and layer hierarchy towards pattern information capture. The term CNN was however used for the first time by Lecun et al. (1998), proposing a model consisting of convolution, pooling and fully connected layers, for handwritten character recognition. However, the model's lack of performance was then subject to limited availability of dataset and depth of architecture. The major breakthrough in CNN architecture came with the availability of ImageNet dataset; Deng et al. (2009), GPUs, and the development of AlexNet; Krizhevsky et al. (2012), which took advantage of parallel compute to process a deep architecture, consisting of 5 convolutional layers and 3 fully connected layers.
With its primary use case in computer vision, the CNNs performance over the years has significantly improved, driven by growing depth, and the use of techniques such as residual blocks to handle vanishing gradient problems, He et al. (2016). The availability of greater computational power has also allowed for training of models over larger datasets, ensuring better generalization and improved accuracy.

A CNN only model consists of 3 basic layers:

- Convolution- The convolutional layer performs a dot product between the learnable kernel matrix and a portion of the receptive field. This kernel slides across the input data to generate an activation map. Further, within the activation map bounds, the convolution carries shared parameters, ensuring that the learned information/weights remain transferable to changes in input.

- Pooling- It reduces the spatial size of the features while retaining the essential information. This is accomplished by dividing the feature map into multiple uniform non-overlapping regions. Each region is assigned a score, calculated commonly as average of that region or maximum value among the features in the region.

- Fully connected- Towards the end of the CNN architecture, the fully connected layer is used to map each of the extracted features form the previous layer to all vectors of the desired output. This helps the model learn output representation based on collective information of features.

While, CNN models are good at capturing spatial dependencies, LSTMs as discussed in section 2.2.2, are good at capturing temporal. Taking advantage of these strengths, a combined architecture CNN-LSTM, first proposed by Sainath et al. (2015), can help model the spatiotemporal dependencies in sequential data.

2.3 Hyperparameters

Hyperparameters are a key towards making a useful machine learning model. Unlike parameters that are internal to the model and learned from the training data, hyper parameters are set prior to the training process and are not a part of the trained model itself. By controlling them the learning behaviour of the model can be adjusted, enabling it to find optimal configuration of parameters that best fit the training data and generalize well over unseen data.

Some of the common hyper-parameters include:

- Train-validation-test split ratio- It determines the portion of data allocated for training, validation and testing purpose. The training set is utilized to learn the model parameters. During the training process, the validation set is employed to assess the performance of the model over unseen data. The results obtained from the validation set guide decisions regarding model architecture, including the selection of hyper-parameter values. Finally, after the training and validation steps, the test set is employed to provide an unbiased evaluation of the model's performance, serving as an indicator of its real-world effectiveness. The test set performance is also utilized as a metric for comparing the efficacy of different model architectures.

- Number of hidden layers- It affects the model's capacity to learn complex patterns. Deepening the network by adding more hidden layers, the potential to capture intricate relationships in the data increases. However, this comes at a trade-off of increased time complexity. Moreover, beyond a certain limit, increasing the depth can lead to overfitting, where the model becomes too specialized over training data and fails to generalize over unseen data. Determining the optimal number of hidden layers requires empirical experimentation and careful consideration of various factors, including the size and complexity of the dataset, as well as the availability of computational resources. While there are no definitive rules for selecting the number of hidden layers, some guidelines have been suggested by Jeffheaton, which include:

- Number of neurons in each hidden layer- Similar to number of hidden layers, the number of neurons is a critical factor that impacts the model's learning capacity. Neurons in the hidden layers are responsible for transforming the input data into higher-level representations and extracting complex patters. However as the number of neurons increase, the number of learnable parameters in the architecture also increases, leading to higher time complexity. This increase can also lead to memorization of noise in the training data, leading to overfitting and poor generalization. The selection of optimal number of neurons is influenced by similar factors as choos-

Table 2.1: Determining the number of hidden layers- Rules of thumb

Number of Hidden Layers	Result
none	Only capable of representing linear separable functions or decisions.
1	Can approximate any function that contains a continuous mapping from one finite space to another.
2	Can represent an arbitrary decision boundary to arbitrary accuracy with rational activation functions and can approximate any smooth mapping to any accuracy.

ing the optimal number of hidden layers. One of the rules of thumb for determine this number is proposed by Jeffheaton.

Table 2.2: Determining the number of neurons in hidden layers- Rules of thumb

Sr no.	Result
1	Should be between the size of the input layer and the size of the output layer.
2	Should be 2/3 the size of the input layer, plus the size of the output layer.
3	Should be less than twice the size of the input layer.

- Dropout rate- Dropout, first proposed by Srivastava et al. (2014), is a regularization technique used to prevent overfitting during the training process. The training of a dropout neural network is about similar to a standard neural network. The only difference being, for each mini-batch in the training epoch a thinned network is sampled by dropping out units. The percentage dropout of units corresponds to the dropout rate. This form of model averaging helps learn more robust representation of weights, making the model predictions less susceptible to individual neurons. However beyond thresholds, an increased dropout can potentially reduce the network's capability to learn complex patterns, while a reduced dropout could provide insufficient regularization and risk the model into overfitting. The optimal dropout rate depends on various factors, including the size and complexity of the dataset, the model architecture etc.
- Number of epochs- It is the number of times the entire dataset will pass through the learning algorithm during the training process. A single epoch flow consists of a three step process:
  - Step 1- Forward pass to compute the output.
  - Step 2- Calculation of gradients by comparing predicted output from the previous step against true target value.
  - Step 3- Backward pass to update the model's parameters according to the calculated gradients from the previous step.

  With this update process, epochs help the model learn through an iterative process. Like other hyper-parameters, optimal number of epochs is dependent on multiple data and model architecture related factors. An insufficient number of epochs can lead to under-fitting while an excess can lead to overfitting. To address this challenge, a common approach involves the use of best model search. In this process, an the model is run over an excessive number of epochs, and the epoch for which the model parameters yield the lowest error/ loss are saved as the optimal configuration. This iterative search helps strike a balance between underfitting and overfitting, enabling selection of an appropriate number of epochs for optimal model performance.

- Loss function- It is also known as object or cost function, and it outputs an evaluation of the model's prediction accuracy. The loss correlates to the deviation between predicted and true target values. When the deviation is smaller then the loss function will yield a smaller value, indicating a better prediction accuracy. Conversely, when the deviation is larger, a larger value is outputted as the loss, indicating higher prediction error. This quantitative evaluation is used as a factor in calculating the update step size for updating the model's parameters during back propagation. It aligns with the training objective of reducing the output value of loss function. Two of the most popular loss functions include:
  - Mean Square Error (MSE)- As shown in equation 2.14, its implementation involves taking the difference between the predictions and the ground truth, squaring it, and then averaging across the entire dataset.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y_i})^2 \tag{2.14}$$

    where, n is the total number of samples, Y_i and $\hat{Y_i}$ represent the true and the predicted target values respectively for the i^{th} sample. MSE loss function is commonly employed in regression problems.
  - Cross Entropy - It is also called logarithmic or log loss. As shown in equation 2.15, a log function is fitted over predicted probabilities of the target classes, making a similarity measure between the predicted and the true probability distribution. The loss function outputs a very large score when the dissimilarity approaches the maximum deviation of 1, and a small score for values close to 0.

$$CE = -\frac{1}{n}\sum_{i=1}^{n} Y_i \log(\hat{Y_i}) \tag{2.15}$$

    where, n is the total number of samples, Y_i and $\hat{Y_i}$ represent the true class and the predicted probability value respectively for the i^{th} sample. CE loss function is commonly employed in classification problems.
- Learning rate- A model's learning process occurs during the backpropagation step, where the model's weights are updated. One popular algorithm used for optimization is stochastic gradient descent, as shown in equation 2.2. In this algorithm and others, the learning rate is a critical hyperparameter that controls the magnitude

of weight updates based on the estimated error. The learning rate determines how quickly or slowly the model adapts to the training data. A too small value may increase the time complexity, requiring greater number of epochs and longer training time. This can pose challenges when working with large datasets and complex models. Conversely, too large learning rate may result in unstable training process with oscillations, and potential convergence to sub-optimal solutions. To address these issues, variable learning rate methods such as adaptive learning rate gradient descent and learning schedule are employed.These approaches aim to strike a balance between convergence speed and quality, by dynamically adjusting the learning rate during training.

- Activation function It is also known as transfer function and is used to compute the model node outputs. As shown in figure 2.2 a node takes a weighted summation of inputs, adds bias to it, and passes through the activation function, producing the output. One of the key reasons of introducing an activation function in the flow process, is to allow for non-linear transformations, enabling the model to learn and predict complex non-linear patterns. Some of the other advantages offered are the ability to normalize and scale the output values, facilitating the output to lie in a certain range, thus ensuring a suitable representation of data for learning by subsequent layers. Some of the popular activation functions include:
  - Sigmoid- It takes a real value as input and outputs a value in the range 0 to 1. Equation 2.16 gives the mathematical representation of the function:

    $$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.16}$$

    where, σ is the sigmoid function mapped over the input x, and e is the Euler's constant with the value 2.781. We can see that as the value of $x \to -\infty$ the function $output \to 0$, and when $x \to +\infty$ the function $output \to 1$. It is commonly used for modelling prediction probabilities.
  - Tanh (Hyperbolic tangent)- It follows a S-shape, similar to the sigmoid activation function, however outputting values in the range of -1 to 1. The function can be represented as in equation 2.17.

    $$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.17}$$

    where, $\tanh$ is the hyperbolic tangent applied over the function input x, and e is the Euler's constant. We can observe that, as $x \to -\infty$ the $output \to -1$, and as $x \to +\infty$ the $output \to 1$. As the output is zero centered, it exhibits symmetry around the origin, proving beneficial in capturing data that carries negative and positive values. Additionally, the steeper gradient of the function compared to sigmoid can lead to faster convergence.
  - Rectified linear unit- As a computationally efficient function, it uses a multi-step linear functions to handle non-linearity. The function output can be represented by the equation 2.18:

    $$f(x) = \max(0, x) \tag{2.18}$$

where, x is the input to the rectified linear unit function f. From equation 2.18 we also observe that the input is sent through as output for values more than zero and 0 for values less than or equal to zero. This introduces a key limitation, the functions' inability to handle and learn the representation for negative values.

- Softmax- It is a combination of multiple sigmoid functions, and converts the input real numbers into a probability distribution. The function can be represented by equation 2.19:

$$softmax(x_i) = \frac{exp(x_i)}{\sum_{j=1}^{n} exp(x_j)} \tag{2.19}$$

where, x_i represents the input value for the i^{th} element of the vector, n is the number of possible classes as outcome, $exp(x_i)$ denotes the standard exponential function applied to x_i, and $\sum_{j=1}^{n} exp(x_j)$ represents the summation of the exponential values over all elements in the vector. By using this formulation, the softmax function normalizes the values and output a probability distribution for the n possible classes. Do its ability to handle multiple classes, it has been a popular choice in classification problems.

• Optimization algorithm- It is a method used during the training process, to iteratively adjust the attributes of the neural network such as weights and learning rate. The algorithm type defines the calculation towards updating the weights and learning rate, in response to loss function output value. Some popular optimization algorithms include:

  - Gradient descent- It is one of the most basic but widely used optimization algorithms. It uses the first-order derivative of the loss function to measure the slope of the function at a given point. The slope is then used to guide the iterative update process of the attributes in the direction of the steepest descent. The gradient descent updates aim to minimize the loss function, and thus enhance the model's performance. Equation 2.20 gives the algorithm implementation.

$$\theta_{t+1} = \theta_t - \alpha \nabla J(\theta_t) \tag{2.20}$$

where, θ_t are the parameter values at iteration step t, α is the learning rate, and $\nabla J(\theta_t)$ is the gradient of the loss function J with respect to θ. Since for a single update step, gradients for the entire dataset need to be calculated, the algorithm can be computationally expensive for large datasets.

  - Stochastic gradient descent (SGD)- It is a computationally efficient variant of standard gradient descent. SGD, represented by equation 2.21, model's parameters are updated more frequently with the computation of loss on each randomly selected training example x_i and label y_i. The stochastic nature, coupled with frequent update steps, facilitates faster convergence of the model while avoiding the issue of getting stuck in local minima.

$$\theta_{t+1} = \theta_t - \alpha \nabla J(\theta_t; x_i, y_i) \tag{2.21}$$

However, due to frequent updates with high variance in parameter values, the objective function can experience heavy fluctuations. To overcome this challenge, one can either reduce the learning rate or utilize the mini-batch variant of SGD, where updates are performed on randomly selected subsets of the training data.

– Adaptive moment estimation (Adam)- Proposed by Kingma and Ba (2014), Adam has been a popular choice as an optimization algorithm for regression problems. It uses the first and second-order moments to maintain two moving averages, represented by equations 2.22 and 2.23.

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.22}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.23}$$

where, the first moment $\hat{m}_t$ is the mean, and the second moment $\hat{v}_t$ is the uncentered variance of the gradients respectively. β_1 and β_2 represent their respective decay rates, and hold constant values. The update step can then be represented by equation 2.24 :

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \tag{2.24}$$

In practice, the constant values of β_1 = 0.9, β_2 = 0.999, and ϵ = 10^{-8} have been found to work well.

The internal parameters, commonly referred to as parameters, are initialized at the beginning of the training and undergo continuous update during the process. The continuous updates are influenced by a set of external parameters known as "hyperparameters". Two of the key learnable, internal parameters include:

- Weights- As also discussed in Section 2.2 and shown in in figure2.2, weight are parameters associated with the connections between different neurons in the neural network. Using a multiplicative function, they determine the importance assigned to the input signals that propagate through these connections, and thus the importance of input received by the neurons. During training the weights are adjusted to reduce the cost function, minimizing the deviation between the predicted and the target output.

- Biases- Shown also in figure 2.2, these are added to the weighted inputs of the neural network. They map the offset between the predicted and intended outputs of the neurons, helping shift the activation function towards the positive or negative side, along the X-axis.

Chapter 3

AIS data

This chapter provides a brief overview of the automatic identification system (AIS), the data provided by the Norwegian coastal administration for this thesis and the pre-processing steps undertaken towards generating the required data-structure.

3.1 AIS

The AIS, is a tracking and navigation assistance system that allows vessels to share their static and dynamic information via a transponder. The transponder receives data from vessels in vicinity while also transmitting own vessel data embedded in VHF based radio signals. The history of AIS can be dated back to March 24, 1989, with the oil tanker Exxon Valdez running aground in Alaska's Prince William Sound, spilling 11 million gallons of crude oil, leading to one of the largest ship sources oil-spill incidents and environmental disasters. The AIS was then developed as a means to improve situational awareness of vessels by providing tracking capabilities for shore based Vessel Traffic Services (VTS), Kimbra Cutlip

Then in 2000, as part of revised SOLAS regulation V/19 - Carriage requirements for ship-borne navigational systems and equipment, AIS was made mandatory by IMO for all passenger ships, international voyage vessels more than 300 gross-tonnage and territorial cargo vessels more than 500 gross-tonnage.IMO; Towards enhanced navigational safety and exemptions of pilotage when operating in coastal waters; Ministry of Transport, AIS has also been widely adopted by many smaller crafts.

3.2 AIS Dataset

For the scenario described, the AIS data was provided by Norwegian Coastal Administration and for the Olso fjord region (Latitude Minimum/Maximum: 60°/58.6°; Longitude Minimum/Maximum: 11.5°/9.4°), as also show in figure 3.1. The identified relevant features of our data-frame are presented in Table 3.1.

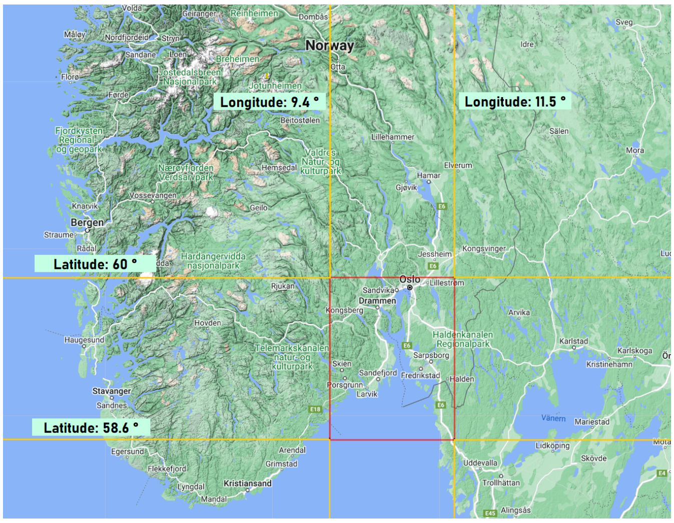


Figure 3.1: AIS dataset region (outlined in red)

Features	Description	Unit
MMSI	Maritime Mobile Service Identity number. - 9 digit number, unique for all vessels. - First 3 digits represent the vessel's flag of registration.	unitless
Unixtime	Timestamp of AIS message - Unix format represents number of seconds that have elapsed since January 1, 1970.	seconds
Latitude	Geographic coordinate (WGS84 format) - Measured in decimal degrees [-90°, 90°]	degrees
Longitude	Geographic coordinate (WGS84 format) - Measured in decimal degrees [-180°, 180°]	degrees
Speed over ground (SOG)	Vessel's speed relative to the earth's surface. - Measured in knots (nautical miles/hour)	knots
Course over ground (COG)	Vessel's course over the earth's surface, measured clockwise relative to true north. - Expressed in degrees (0° to 360°)	degrees

Table 3.1: AIS dataset features under consideration

3.3 Raw dataset

Some of the specifics of the raw dataset are provided in Table 3.2. It's worth noting that the dataset for the year 2019 contains about 305 million data-points, making it computationally heavy to handle as a single file. So, for ease the datasets were split according to months. Figure 3.4 visualizes the unique MMSI counts for each month, showing peaks during the tourism seasons. The AIS message time intervals for the given data are explored using plots as shown in figures 3.2 and 3.3. From the former we can see that majority of the vessels have an average AIS transmission frequency in the range of 0 to 120 seconds.

Further in the later plots, we see that the majority of the AIS messages carry a transmission frequency in a similar range, 0 to 120 seconds. The month of January shows a similar trend. This time interval trend has further been used as a basis for "voyage identification" in the data processing stage.

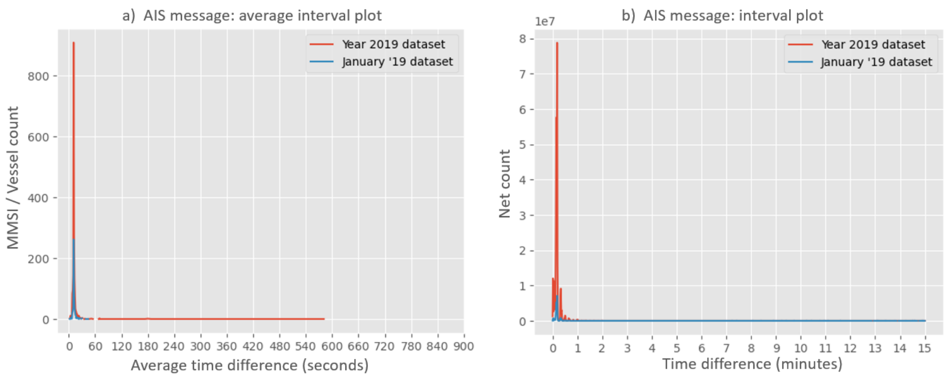


Figure 3.2: a)Average time difference vs unique MMSI count. b)Time difference vs AIS message count (Year 2019 AIS messages)

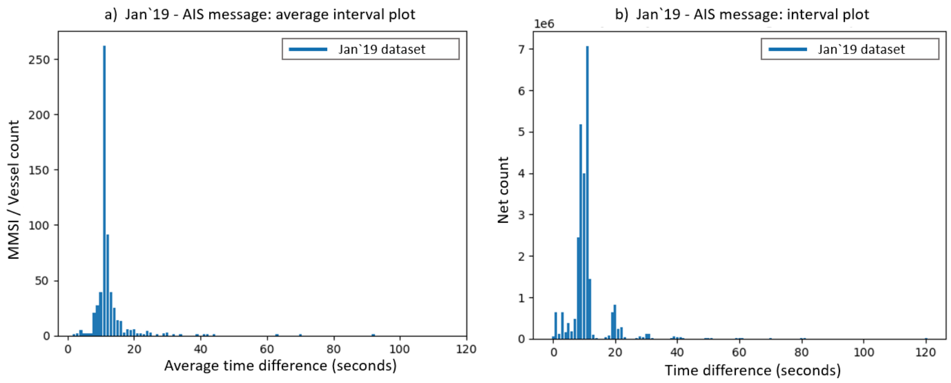


Figure 3.3: a)Average time difference vs unique MMSI count. b)Time difference vs AIS message count (Jan‘19 AIS message)

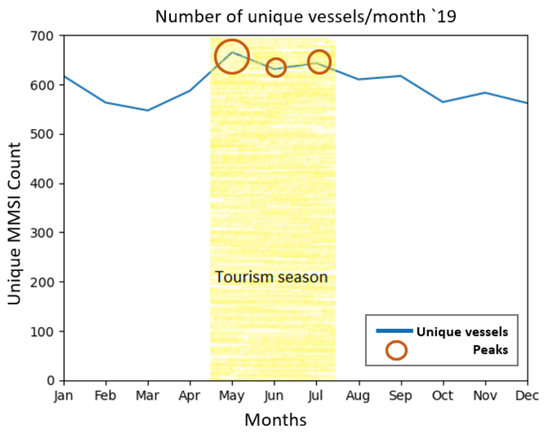


Figure 3.4: Number of unique vessels per month distribution (Year 2019)

Raw Dataset	
Properties	**Details**
Location	Oslo fjord region - Latitude max/min: 60°/58.6° - Longitude max/min: 11.5°/9.4°
Time period	2019 (1st January 2019 - 31st December 2019)
Shape (without duplicate rows)	304892534 x 10 - (∼304 million rows and 10 columns)
Duplicate rows	1350083
Key Features	**Unique counts**
MMSI	2273
Unixtime	22936229
Latitude	792662
Longitude	1205199
Speed over ground (SOG)	913
Course over ground (COG)	3624

Table 3.2: Raw AIS Dataset specifics

The raw data-structure can be represented as:

$$df_{raw} = [r_1, r_2, r_3, ..., r_n] \tag{3.1}$$

where r represents each row of the AIS message, given as:

$$r_{index} = [MMSI_i, t_j, P_{ij}, h_{ij}, v_{ij}] \tag{3.2}$$

where $MMSI_i$ is the MMSI number of a vessel, t_j is the time stamp, and P_{ij}, h_{ij} and v_{ij} are the position vector, COG and SOG respectively for the i^{th} vessel at j^{th} timestamp. The position vector can further be written as:

$$P_{ij} = [\phi_{ij}, \lambda_{ij}] \tag{3.3}$$

where ϕ_{ij} and λ_{ij} are respectively the latitude and longitude coordinates in degrees, WGS84 format.

3.4 Data preprocessing

The effective processing of the raw data is a key towards improving the performance of any Machine Learning model. Towards same, a multi-step data processing framework was implemented as illustrated in figure 3.5 and later discussed in Section 4.1.

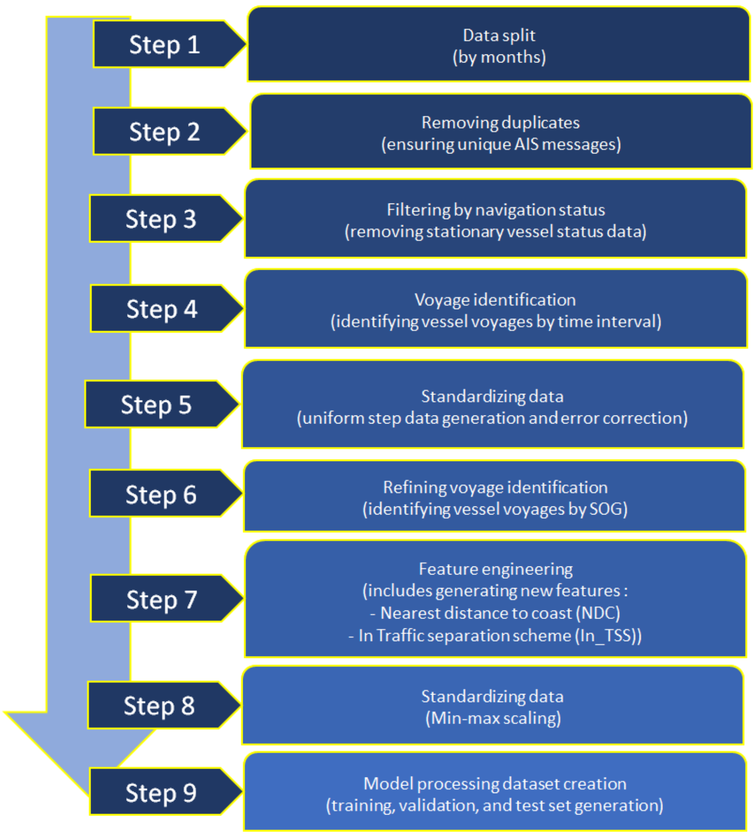

Step 1
Data split (by months)
Step 2
Removing duplicates (ensuring unique AIS messages)
Step 3
Filtering by navigation status (removing stationary vessel status data)
Step 4
Voyage identification (identifying vessel voyages by time interval)
Step 5
Standardizing data (uniform step data generation and error correction)
Step 6
Refining voyage identification (identifying vessel voyages by SOG)
Step 7
Feature engineering (includes generating new features : - Nearest distance to coast (NDC) - In Traffic separation scheme (In_TSS))
Step 8
Standardizing data (Min-max scaling)
Step 9
Model processing dataset creation (training, validation, and test set generation)

Figure 3.5: Multi-step data processing framework

3.4.1 Libraries for data processing

The libraries used as part of the data processing in python were:

- pandas; Wes McKinney (2010) for data handling
- NumPy; Harris et al. (2020) for array creation and manipulation
- Matplotlib; Hunter (2007) for data plotting and visualization
- math; Van Rossum (2020) and SciPy; Virtanen et al. (2020) for mathematical calculations and interpolation
- datetime for handling date-time stamps
- GeoPandas; Jordahl et al. (2020), haversineBalthazar Rouberol and vincenty; Maurycy Pietrzak for handling latitude, longitude data and performing geospatial calculations

3.4.2 Data processing steps

Showing in figure 3.5, the sequential steps involved in the processing of data are:

- Step 1- Data split
  As discussed in section 3.3, the 2019 year raw data was split by months to allow for ease of handling files. With this partitioning, 2GB of memory was occupied against a month's dataset, compared to the earlier 24 GB requirement for the entire year.

- Step 2- Removing duplicates
  This involved dropping duplicates from the dataset to ensure only unique AIS messages were present.

- Step 3- Filtering by navigation status
  Next, towards having only movement trajectories as a part of modeling problem, we filter out AIS messages by navigation status, where 70°of the vessels average SOG is either less than 2 knots or is intended to be zero. 1(anchored), 5(moored), 6(aground), 7(fishing) and 13 were identified for filtering.

  From figure 3.6 and 3.7, it is observed that while the threshold line for anchored and fishing vessels is at less than 2 knots, moored vessel have the 70°threshold line above 5 knots. This indicates a wide disparity between the recorded navigation status and SOG of a large number of vessels, suggesting potential non-updated/incorrect state information in the AIS. With its 50°threshold line at around 2 knots, the navigation class type was taken into preference and AIS messages belonging to 5(moored) class were removed. Figure **??** shows AIS messaged relative distribution prior and post filtering, with state 0 (Underway using engine) accounting for more than three-quarter of the messages post filtering.

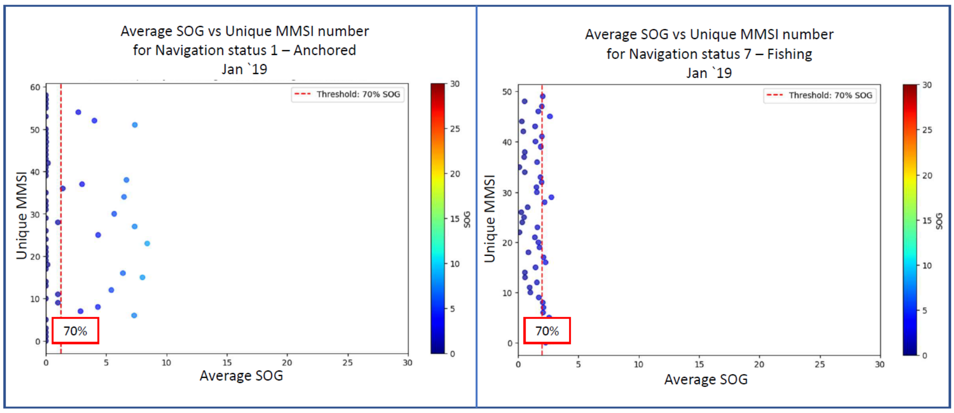


Figure 3.6: Average SOG vs Unique MMSI number for navigation status a)1-Anchored (Left), b)7-Fishing (Right)

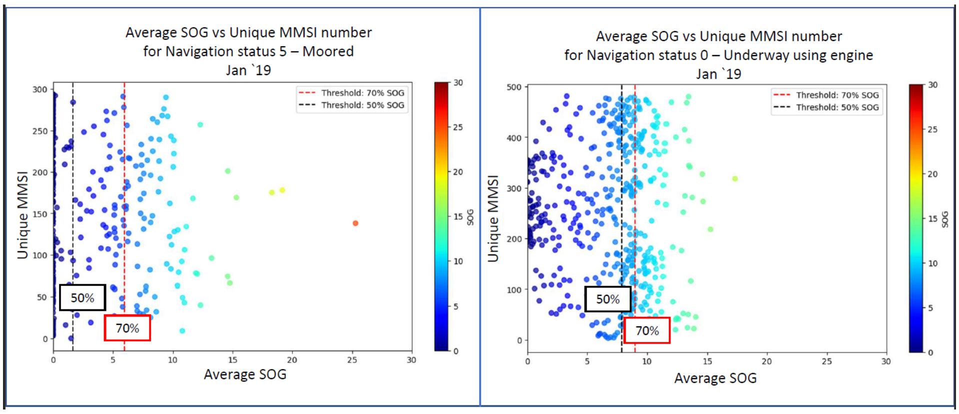


Figure 3.7: Average SOG vs Unique MMSI number for navigation status a) 5-Moored (Left), b) 0-Underway using engine(Right)

- Step 4- Voyage identification As seen from figures 3.2 and 3.3 (in section 3.3), the average frequency of AIS messages transmission falls within 120 seconds for majority of the vessels. Taking a factor of five, we consider 10 minutes as our threshold for voyage identification. Consequently, we assign a unique voyage id to a vessel if the time difference between consecutive signals exceeds the threshold. Figure 3.8 illustrates that the majority of the vessels executed between 0 to 20 voyages in January'19.

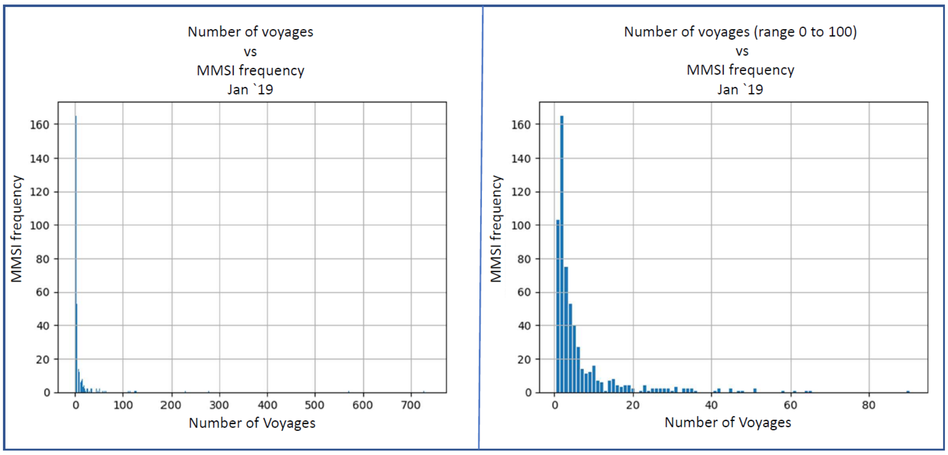


Figure 3.8: Number of voyages identified vs corresponding MMSI frequency a) entire range of number of voyages b)number of voyages in the range 0 to 100

- Step 5- Homogenizing data
  To ensure equal intervals between timestamps in the dataset, a combination of up-sampling and down-sampling techniques is utilized. This homogenization process is crucial for multi-step prediction problems as it helps the model learn temporal patterns and enhances prediction reliability. As a part of up-sampling interpolation techniques combining bearing and haversine distance, given in equations 3.4 to 3.9 are used for geographic co-ordinate estimation (latitude-longitude) and linear interpolation, given in equation 3.10 for COG. The dataset is then down-sampled at a 30 second message frequency. SOG estimations are then carried out for each of the down-sampled timestamps using the haversine distance between consecutive message and the elapsed time-interval. Other than for training, the SOG feature plays an important role in the subsequent data processing step.

$$a = \sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1)\cdot\cos(\phi_2)\cdot\sin^2\left(\frac{\Delta\lambda}{2}\right) \tag{3.4}$$

$$c = 2\cdot \text{atan2}\left(\sqrt{a}, \sqrt{1-a}\right) \tag{3.5}$$

$$d = R\cdot c \tag{3.6}$$

$$X = \cos\phi_2 \cdot \sin\Delta\lambda \tag{3.7}$$

$$Y = \cos\phi_1 \cdot \sin\phi_2 - \sin\phi_1 \cdot \cos\phi_2 \cdot \cos\Delta\lambda \tag{3.8}$$

$$\beta = \text{atan2}(X, Y) \tag{3.9}$$

where,
for two co-ordinates in radians (ϕ_1, λ_1) and (ϕ_2, λ_2), $\Delta\phi$ is the difference in latitude radians, $\Delta\lambda$ is the difference in longitude radians, a is the haversine of angular distance between two points on a sphere, c is the central angle in radians, R is the radius of the sphere (earth; 6371 km), and d is the distance computed between two points, and β is the bearing angle in radians.

$$H = h_1 + \left(\frac{h_2 - h_1}{t_2 - t_1}\right) \cdot (T - t_1) \tag{3.10}$$

where h_1 and h_2 are the given values of COG at timestamps t_1 and t_2 respectively, T is interpolation target timestamp and H is the the corresponding COG to be interpolated.

- Step 6- Refining voyage identification
  Bar plot in figure 3.9(a) shows the distribution of MMSI values and their percentage AIS messages that carry the SOG value as 0 knots (pink), 0 or 1 knots (grey), and 0,1 or 2 knots (yellow). It can be observed that in a significant portion of the dataset constituting many MMSIs have a majority as stationary vessel states. This makes the dataset complex for learning trajectory motion predictions, as the lack of vessel movement introduces noise into the sets of trajectory patterns and thus the modeling problem.

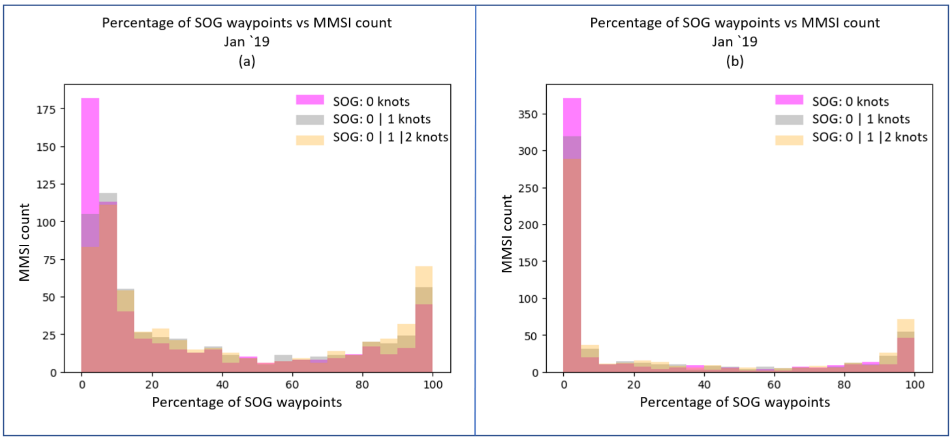


Figure 3.9: Percentage of SOG way-points vs MMSI count- January'19 a) without SOG based processing b) after handling voyage start and end points

Towards overcoming this constraint in the dataset, SOG based processing steps are applied:

– Step 1- Refining voyage start and end points
  In this step, the SOG information is used to determine the beginning and the end of each voyage. In case of multiple zero SOG points at the beginning or/and end of a voyage, only a single zero SOG point is appended to allow for a zero-SOG-start or/and zero-SOG-end. The output of this step is reflected in figure 3.9(b), where it can be observed that the MMSI counts carrying lower SOG values has significantly reduced.

– Step 2- Refining voyage identification
  As a part of its implementation a two step process is carrid out. First, consecutive zero SOGs are used to split voyages into sub-voyage sets. Next, these sets are processed to filter out sub-voyages that contain all values of SOG as zero. Figure 3.10 (b) shows the output of this step, reflecting a further increase in MMSI counts that carry lower percentages of zero SOG values.

– Step 3- Voyage selection
  As a final step in the SOG based data processing, voyage selection is carried out based on voyage duration threshold. To align with our modeling problem, a threshold of 20 timestamps (equivalent to 10 minutes) is used to identify suitable voyages for the dataset.

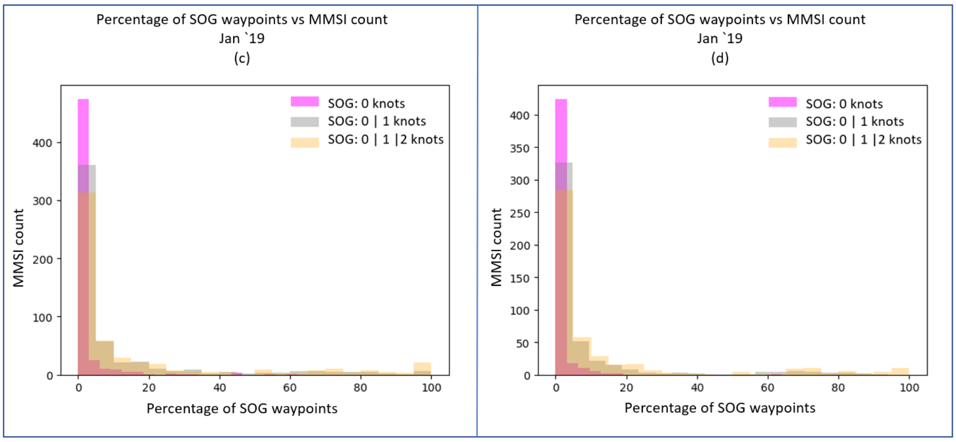


Figure 3.10: Percentage of SOG way-points vs MMSI count- January'19 a) after handling voyages b) After keeping voyages that last at least 20 timestamps (i.e. 10 minutes)

- Step 7- Feature engineering In a maritime use case of coastal waters, the trajectory of a vessel often depends on vessel-specific characteristics, such as length, draft, distance to coast, propulsion power, maneuverability, rule-base restrictions, and many other factors. In order to accommodate some aspects of these additional characteristics into our dataset, feature engineering was performed. The created features were:

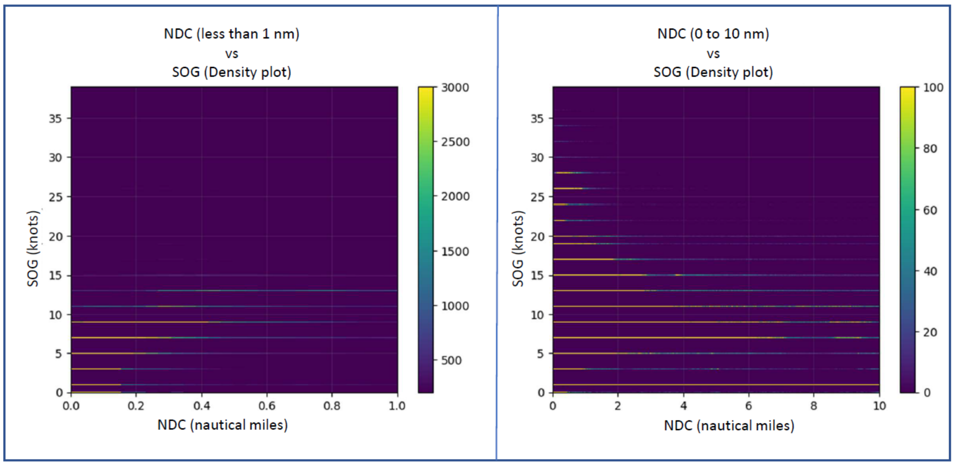


Figure 3.11: NDC vs SOG distribution a) NDC in range (0 to 1) nm b) NDC in range (0 to 10) nm

– Nearest distance to coast (NDC):
It was incorporated to provide a generalized geographic feature. While latitude and longitude provide location specific characteristics, the NDC allows for capturing generic dynamic properties of a vessel in relation to its proximity to coastline. One such dependence can be seen from figure 3.11(a), where the SOG values tend to increase with an increase in NDC. However from Figure 3.11(b) it is also observed that, as the distance from coast increases beyond a range, slower vessels dominate movement patterns. This is attributed to the fact that majority of the smaller and more dynamic vessels execute voyages near coastal boundaries.NDC was engineered using two datasets, one from the upstream data processing, and other was the Norwegian coastline dataset, available open source by National Oceanic and Atmospheric Administration (NOAA).

Table 3.3: Bounding Box: Oslo region dataset coastline

Direction	Coordinates (Latitude, Longitude)
Northwest	(59.93, 9.4)
Northeast	(59.93, 11.45)
Southeast	(58.6, 11.45)
Southwest	(58.6, 9.4)

First the coastline data for the Oslo fjord region is extracted by defining a bounding box, with limits as shown in table 3.3. This bound box encompasses the region of interest as per the available AIS data. Subsequently, R-trees are constructed to make the nearest search computationally simple. Finally, a nearest neighbour search is performed using the R-tree index, and the nearest distance to coast calculated. The haversine distance is used as a measure to both, compare and calculate the nearest distance to coast, distance between the vessel co-ordinate and the closest coastline co-ordinate.

– Vessel belonging to Traffic separation scheme (TSS_Vessel) and Vessel in Traffic separation scheme (In_TSS):
In confined waters the concept of TSS is used to establish safe and efficient vessel movement. The dataset used in this thesis belongs to the Oslo fjord region, where all vessels longer than 24 meters have to apply the TSS established in the region. The TSS is divided into northbound and southbound lanes, and is about one nautical mile in width.

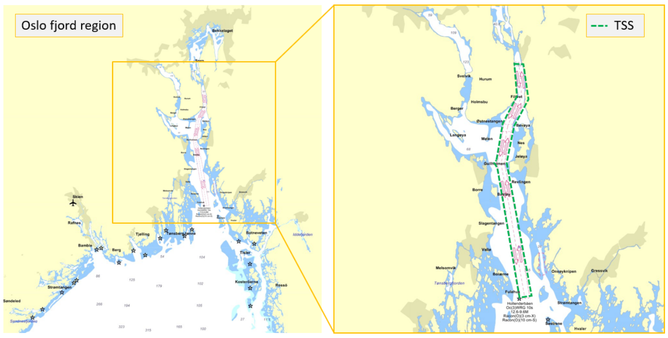


Figure 3.12: TSS map for the Oslo fjord region (highlighted in green)

Figure 3.12 presents the TSS map for the Oslo fjord region, sourced from AS (Year). The designated TSS area is highlighted in green, and is used in conjunction with the AIS dataset to determine whether a vessel is situated within the TSS region (i.e. In_TSS is True) or is outside (i.e. In_TSS is False). Additionally, the length of a vessel is a key property as it often co-relates with the size of of a vessel and thus influences various trajectory related operational constraints, such as speed limitations, allowable draft, maneuverability etc. It also determines the eligibility of a vessel to use the TSS, making it a crucial property with implications on its trajectory. To address this, we identify vessels which traverse the entire TSS passage, categorizing them as vessels belonging to TSS (i.e. TSS_Vessel is True). In a real-world scenario AIS messages can often hide/ be misinformative about vessel properties such as length; United States Coast Guard, consequently, the use of visual aids such as cameras and computer vision techniques can be employed to determine the vessel length; Imber et al. (2019), enabling the classification of a vessel as belonging to TSS or not.

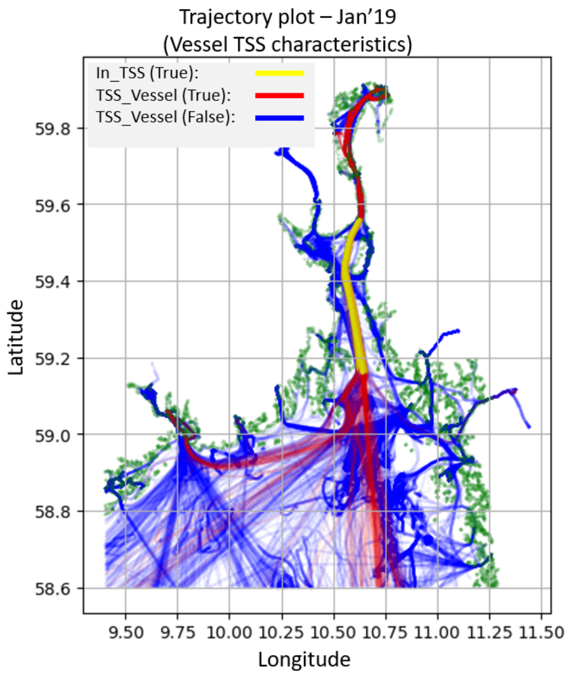


Figure 3.13: Trajectory plot: Vessel TSS characteristics

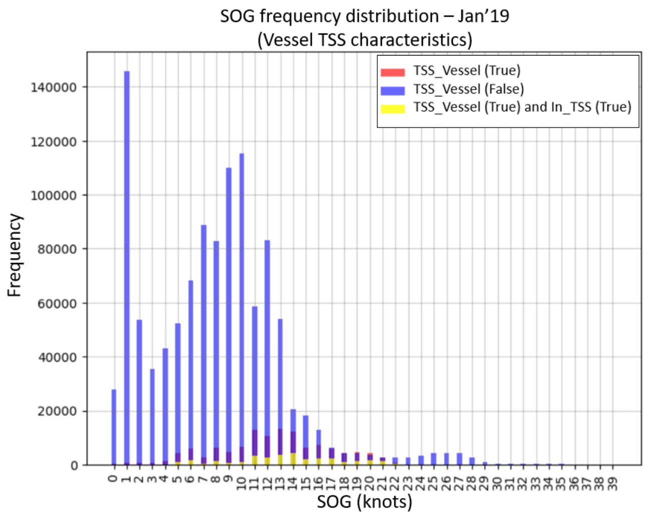


Figure 3.14: SOG distribution: Vessel TSS characteristics

Visual illustrations capturing the properties In_TSS and TSS_Vessel are shown in figure 3.13. In this representation, blue lines represent vessels located outside the TSS, red lines signify vessels belonging to TSS, and yellow lines denote the vessel trajectory section situated within the TSS. From the visual analysis, it is evident that TSS_Vessels in red, tend to follow more linear trajectories with minimal deviations. Further, figure 3.14 also reveals a wider span of velocities for vessels that do not belonging to TSS.

- Step 8- Normalizing data As discussed in section 2.2, machine learning algorithms like neural network often use gradient descent as an optimization technique to gain convergence. From the back propagation equation 2.2, it can be seen that the size of gradient descent is affected by the error size, and thus by the size of feature value. In our specific scenario, the feature magnitudes can exhibit substantial variation. For instance the cog values ranges from 0 to 360 degrees, while the geographic co-ordinates span over a mear 2 degree range. To prevent a bias towards singular feature, and ensure smooth convergence towards minima, normalization becomes a necessity.

  Normalization is a scaling method, which involves re-scaling feature values to a common scale, ranging [0,1]. This Min-Max scaling is implemented using equation 3.11.

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3.11}$$

  Key features for training the model were identified and selected, as listed in table 3.4 With the exception of cog, all other features were normalized to the scale of [0,1]. To preserve the angular information contained in COG and prevent the discontinuity that occurs at 360 °in a circular scale, the COG values were converted into sine and cosine components, spanning a range of [-1,1]. This enables the effective utilization of the cog feature in the model training process.

Table 3.4: Key features and their value range

Sr no.	Key features	Non-normalized values (Min / Max)
1	Latitude	58.6 / 59.3 degrees
2	Longitude	9.4 / 11.4 degrees
3	SOG	0 / 40 knots
4	COG	0 / 360 degrees
5	NDC	0 / 30 nm
6	In_TSS	0 or 1
7	TSS_Vessel	0 or 1

- Step 9- Dataset creation As a final step in data processing and generation phase, train, test and validation sets were created for the machine learning models. First, the normalized data was grouped based on voyage IDs to isolate each voyage. Then, to enable prediction of 5 minutes into the future based on 5 minutes of seen trajectory states, the data was indexed to generate arrays. Each array representing a sub-voyage consisted of 20 timestamps, the first set from index [0,20], and consequently incrementing both the start and end indices by 1, until the end index reaches the last index of the voyage. By structuring the data in this manner, each sub voyage array set represents a trajectory of total 10 minutes duration.

To enhance the realism of predictions to a real-world problem, and prevent over-fitting, a temporal criteria based on months is adopted when creating distinct train, validation and test datasets. A ten-month period from January'19 to October'19 is allocated for training the models. The 11th month, November'19 was assigned as the validation set, and lastly the 12th month, December'19 served as the test set.

Chapter 4

Methodology

This chapter is divided into two sections. The first section addresses the necessity for improved representation techniques and introduces the fundamental principle underlying the developed Relative Dependency Analysis (RDA) approach. The second section provides an overview of the architectures employed for the prediction scenario, detailing the Machine Learning (ML) models utilized.

4.1 The RDA approach

As discussed in the literature review section 2.1, most of the existing AIS-based ship trajectory prediction models employ latitude-longitude coordinates as the target outputs, learned over position and other key features.Figure 1.3 in Section 2.1 illustrates the frequency plot of standardized geo-coordinates. It is evident that even in a standardized form, latitude-longitude coordinates exhibit a large but sparse vocabulary.

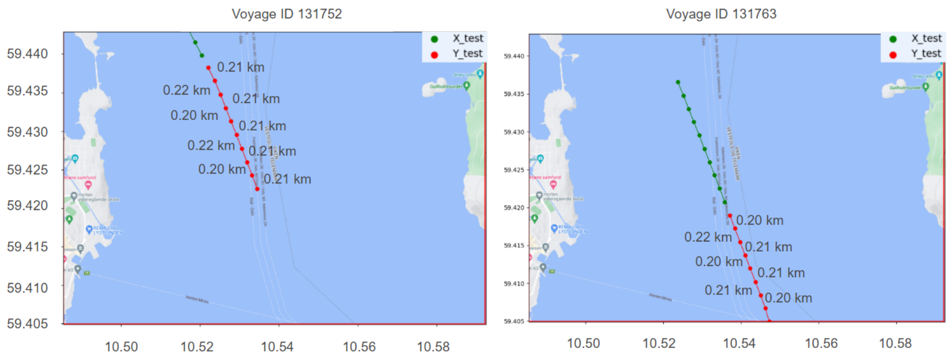


Figure 4.1: Similar trajectory set (Case 1)

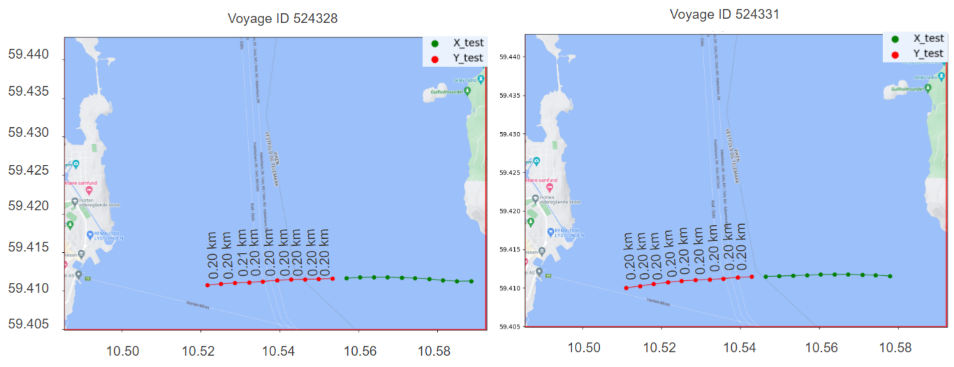


Figure 4.2: Similar trajectory set (Case 2)

In the context of presented figures 4.1 and 4.2, it can be noted that two vessels located at different latitude and longitude positions may exhibit similar movement patterns if their corresponding Speed Over Ground (SOG) and relative Course Over Ground (COG) values remain relatively constant. In such scenarios, when the model's representations are trained to predict latitude-longitude coordinates as key state information, the learned representation for the two cases becomes independent of each other. Consequently, it becomes challenging for the model to learn from similarity and effectively capture the relationship between future position states and observed SOG and COG values. It is important to emphasize that this observed similarity does not imply the lack of usefulness of latitude-longitude position coordinates. The usefulness of geo-coordinates to learn dependence can be observed from figure 4.3, where non-linearity in confined waters would also be a function of geo-coordinates in many cases. Therefore, it becomes crucial to learn the dependence of future positions based on latitude-longitude locations while also considering the high co-relation of future states based on observed COG and SOG states.

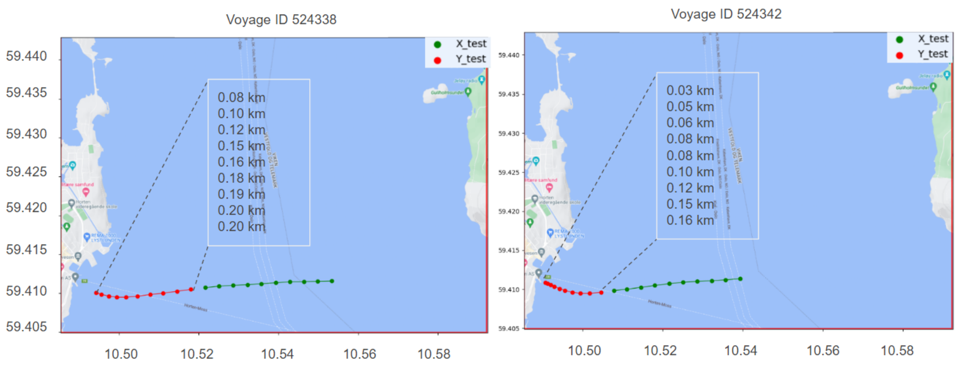


Figure 4.3: Non-linear trajectory set

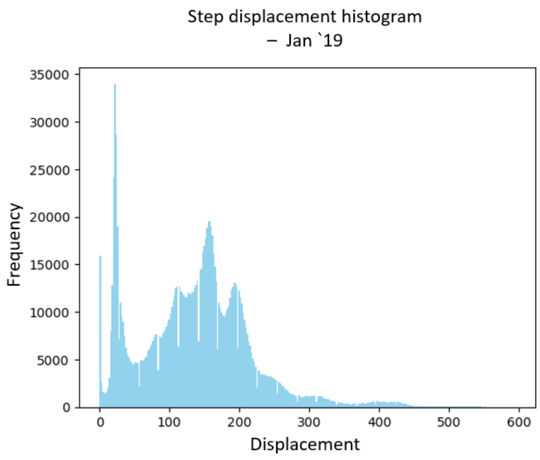


Figure 4.4: Step displacement histogram (Jan'19 processed dataset)

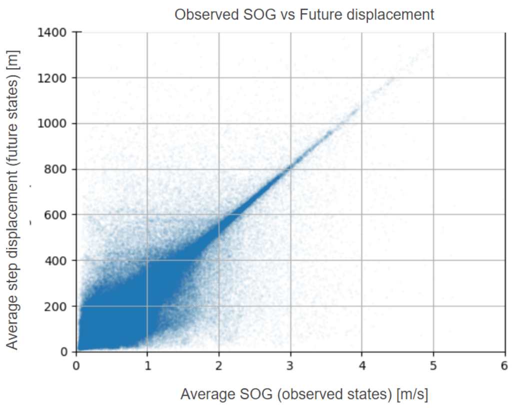


Figure 4.5: Average target step displacement vs average observed state SOG (Jan'19 processed dataset)

To address these challenges, a novel approach called Relative Displacement Angle (RDA) is developed. The underlying concept of this approach is to create a learnable dense representation of data that minimizes the required vocabulary while maximizing accuracy. The histogram in Figure 4.4 indicates that vessel position displacement falls within a specific range, and follows a density pattern. Moreover from plot in figure 4.5 we can observe a high correlation between displacement in future/target states to the SOG in past/observed states. These attributes indicate that displacement can serve as a favorable target feature for accurately mapping vessel positions, given past states.

The implementation of the RDA approach involves a mulit-step preprocessing framework, encompassing steps 1 to 7, as illustrated in Figure 3.5. In step regression analysis there can broadly be two prediction methodologies: Recursive and Direct. A direct approach is adopted to avoid error propagation as the prediction steps progress. With this, the objective of the model is to perform multi-step predictions, where all future trajectory steps are predicted simultaneously and independently. In order to achieve this objective while making use of displacement dynamics, the last input step is used to map the relation between observed state data and the target steps

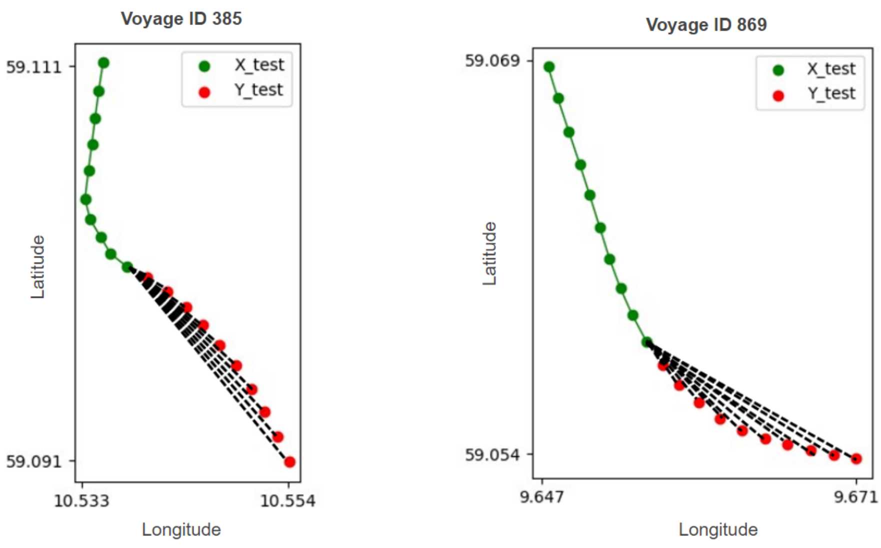


Figure 4.6: RDA relation maps

Furthermore, figure 4.6 provides a visual understanding of the mapped relations in the data, showcasing the patterns that emerges from the link between the last observed state and all future states. Earlier used as a part of data processing and discussed in Section , haversine equations 3.4, 3.5 and 3.6 can be used for calculating the relative displacement between the respective co-ordinates. Additionally,the relative angle is calculated as the difference between the COG of the last observed state and the bearing of the target state, given by equation 4.1.

$$\Delta H_k = h_9 - \beta_k \tag{4.1}$$

where, ΔH_k represents the relative angle, h_9 is the last observed SOG and β_k is bearing of the vector joining the 9^{th} to k^{th} position. The bearing is further calculated using the equations 3.7, 3.8 and 3.9 discussed in section 4.1. The two calculated attributes now serve as the target features for the ML model. The relative displacement and angle predictions can be mapped back to the target geo-coordinates using the reverse haversine analysis. This reverse mapping process involves incorporating the predicted displacement and angle, along with the reference geo-point, in order to obtain the final predicted geo-coordinates for the vessel's position.

4.2 ML Models

This section gives an overview of the employed trajectory prediction models. The architectures used for these models is guided by the theory discussed in section 2.3. Python based libraries like tensorflow have been used for modelling, and numpy, pandas and matplotlib for handling and visualizing datasets and outputs. The 3 models used are:

4.2.1 LSTM

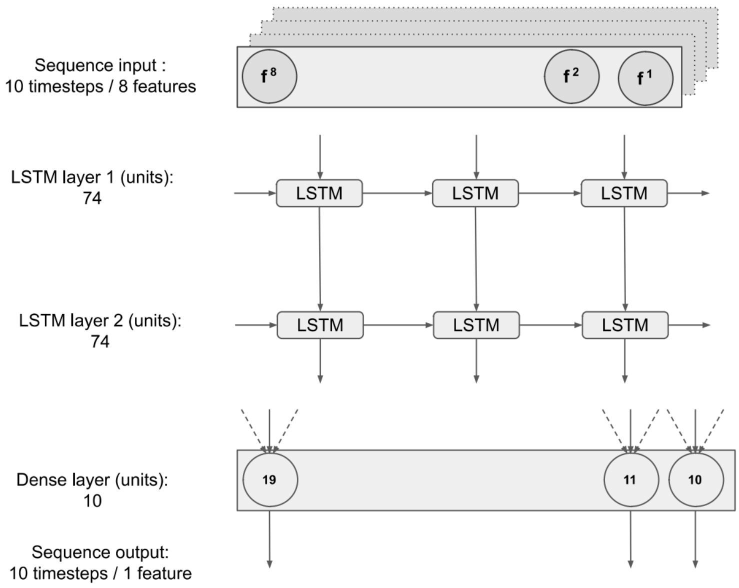


Figure 4.7: LSTM architecture based model

The LSTM cells serve as the primary components of the architecture, supplemented by a dense layer, transforming the LSTM outputs into desired target feature values. Figure 4.7 illustrates the LSTM based model employed with the RDA approach. It uses an input sequence set of dimensions (10x8). Here 10 is the number of time steps and 8 represents the number of features. These sequences are fed to the LSTM layer, consisting of 74 units. The initial LSTM layer processes the input and produces outputs for the second LSTM layer, again consisting of 74 units. These then undergo processing via the dense layer, generating target feature predictions.

To train the model, "MSE" is employed as the loss function. Additionally, as a part of the training process, which includes back-propagating and updating, the "Adam" optimizer is utilized towards optimizing the model parameters, with a learning rate of 1e-3.

4.2.2 GRU

An architecture similar to the LSTM-based model illustrated in Figure 4.7 is developed for the GRU-based model. Instead of LSTM cells, GRU cells are used while maintaining the same number of units and layers at 74 and 2, respectively. As discussed in section 2.2.3, the theory behind GRU, the utilization of GRU cells results in a reduction of approximately 25% in the total number of parameters compared to the LSTM model. This reduction is attributed to the simpler gate structure and consequently fewer parameters associated with the GRU cell. Again, to train the model, “MSE” is employed as the loss function, with “Adam” as the optimizer, and a learning rate of 1e-3.

4.2.3 CNN-LSTM

CNN-LSTM architecture consists of convolution layer, followed by the LSTM architecture consisting of two LSTM layers and one dense layer. The activation function used in this architecture are “ReLU” for the convolution layers and “tanh” for the LSTM layers. In order to generate a comprehensive performance comparison, the training hyper-parameters are kept consistent with those of the LSTM and GRU models, taking “MSE” as the loss function, “Adam” as the optimizer and the learning rate as 1e-3.

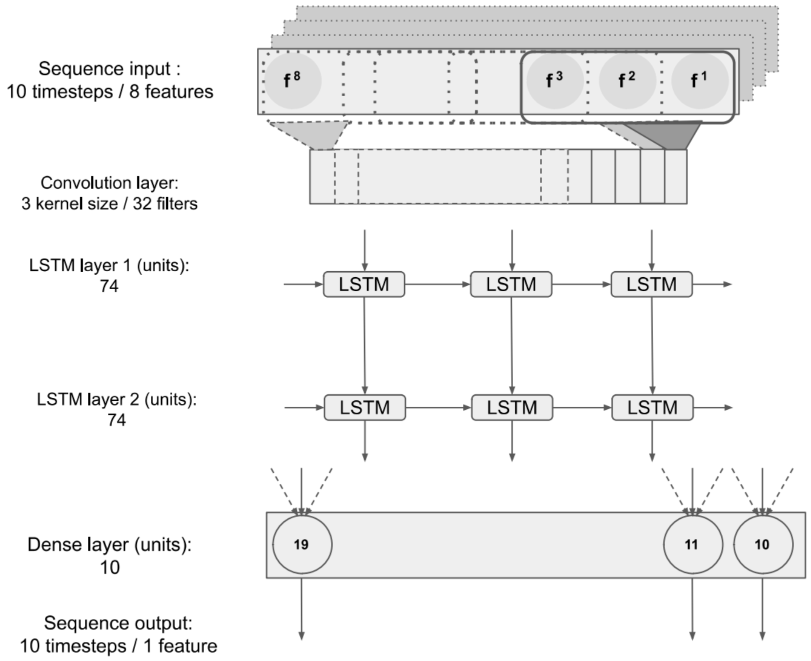


Figure 4.8: CNN-LSTM architecture based model

Chapter 5

Results

This chapter presents the findings derived from prediction tests conducted on the developed models for short-term trajectory prediction in confined waters. The chapter is organized into four main sections, of which three of them focus on the different RNN models. The evaluation and comparison of these models are based on the type of data representation approach utilized. The last section establishes cumulative model comparison, helping understand the strengths and limitations of each model.

5.1 LSTM

The comparative analysis of the LSTM models is presented in table 5.1, which highlights the architecture and accuracy scores achieved using two different data representation approaches. As can be observed that the architectures and hence the number of learnable parameters are very much similar, except for the LSTM-RDA(a) which has fewer parameters due to more compact dense layer with 150 learnable connections compared to 750 in the other layers. Further, it can be observed that the optimal epoch for all four models remains at 20 epochs, the maximum limit, thus indicating and striking a balance between continuous learning and avoiding overfitting. Increasing the number of epoch may lead to further learning but has been capped at 20 for computational efficiency.

When considering the mean squared error (MSE) on the test set, it is important to note that the comparison between models may not be accurate due to the differences in target features and their distribution. Instead, a more reliable measure for comparative analysis is the deviation between the true and predicted geographical coordinates, which is a function of distance. By assessing the deviation in prediction positions, we can gain insights into the effectiveness of the models in capturing the complexities of ship trajectory prediction. This measure takes into account the differences in learned features for the different models and provides a more meaningful basis for comparison.

Table 5.1: LSTM model performance comparison

LSTM architecture				
Models	LSTM-lat	LSTM-lon	**LSTM-RDA(d)**	**LSTM-RDA(a)**
Number of hidden layers	2	2	2	2
Size of input layer	10*8	10*8	10*8	10*8
Size of output layer	10*1	10*1	10*1	10*2
Ideal number of neurons ** ROT[2.2]	73.3	73.3	73.3	73.3
Number of neurons (1st layer)	74	74	74	74
Number of neurons (2st layer)	74	74	74	74
Number of parameters	69,422	69,422	69,422	68,822
Number of epochs	20	20	20	20
Learning rate	0.0001	0.0001	0.0001	0.0001
Lowest error epoch	20	19	20	19
Test set error (MSE)	2.62E-06	3.38E-06	0.000385	0.0398
Mean Displacement Error (nm)		0.094		0.054
Median Displacement Error (nm)		0.066		**0.022**
Final Displacement Error (nm)		0.094		**0.054**

When analyzing the median displacement error, LSTM-RDA exhibits a significantly lower error compared to LSTM-lat and LSTM-lon. The median displacement error of LSTM-RDA is approximately 66% lower than that of the conventional approach. This indicates that LSTM-RDA is more capable of capturing trajectory patterns effectively, leading to superior performance over the conventional LSTM models.

It is also worth noting that when considering multi-step prediction, the deviation errors remain consistent across the prediction steps. This consistency is reflected in the final displacement error and mean displacement error, and is a result of the absence of error propagation over the steps. The alignment between these measures further reinforces the reliability and accuracy of LSTM-RDA in short-term ship trajectory prediction.

In conclusion, the comparative analysis of LSTM models reveals that the architecture and choice of target features significantly impact the accuracy of short-term ship trajectory prediction in confined waters. The LSTM-RDA(d) model, which incorporates relative displacement and angle information, demonstrates superior performance in terms of lower displacement errors. This highlights the importance of considering the spatial relationships between ship positions when predicting future trajectories.

5.2 GRU

The GRU models are evaluated and compared based on their architecture and performance metrics, as presented in Table **??**. The architectures of the GRU models are similar to the LSTM models, with the replacement of LSTM cells with GRU units. The GRU units have a simpler gate structure compared to LSTM, which leads to a lower number of parameters in the GRU models. In fact, the number of parameters in the GRU models is approximately 25% lower than that of the LSTM models. This reduction in the number of parameters implies that the GRU models have significantly lower computational complexity compared to the LSTM models. This makes the GRU models more computationally efficient and suitable for applications where computational resources are limited.

Similar to the LSTM models, the optimal epoch for the GRU models is observed to be around 20. This indicates that the models have reached a stable state of learning and strike a balance between continuous learning and avoiding overfitting.

Comparing the models within table 5.2, it can be observed that GRU-RDA model achieves a significantly lower error compared to GRU-lat lon. The median displacement error of GRU-RDA is approximately 65% lower than that of the conventional approach. This indicates that GRU-RDA is more effective at capturing trajectory patterns and exhibits superior performance in short-term trajectory predictions.

Table 5.2: GRU model performance comparison

GRU architecture				
Models	GRU-lat	GRU-lon	**GRU-RDA(d)**	**GRU-RDA(a)**
Number of hidden layers	2	2	2	2
Size of input layer	10*8	10*8	10*8	10*8
Size of output layer	10*1	10*1	10*1	10*2
Ideal number of neurons ** ROT[2.2]	73.3	73.3	73.3	73.3
Number of neurons (1st layer)	74	74	74	74
Number of neurons (2st layer)	74	74	74	74
Number of parameters	52,698	52,698	52,698	52,098
Number of epochs	20	20	20	20
Learning rate	0.0001	0.0001	0.0001	0.0001
Lowest error epoch	20	19	20	19
Test set error (MSE)	2.09E-06	2.82E-06	0.000384	0.0405
Mean Displacement Error (nm)	0.092		0.056	
Median Displacement Error (nm)	0.063		**0.022**	
Final Displacement Error (nm)	0.092		**0.056**	

5.3 CNN LSTM

Table **??** presents the architecture and performance metrics of the differrent CNN-LSTM model, including CNN-LSTM lat, CNN-LSTM lon, CNN-LSTM-RDA(d), and CNN-LSTM-RDA(a). The CNN-LSTM models incorporate both convolutional and LSTM layers to capture spatial and temporal dependencies in the data. The number of convolutional layers is set to 1, with a kernel size of 3 and a filter size of 32. The number of hidden LSTM layers is 2.

Comparing the performance metrics within Table **??**, it can be observed that the CNN-LSTM-RDA model achieves a significantly lower median displacement error compared to the conventional approach. The median displacement error of CNN-LSTM-RDA is approximately 64% lower than that of the conventional approach. This indicates that CNN-LSTM-RDA effectively captures trajectory patterns and outperforms the other models in short-term trajectory predictions.

It is also worth noting that the CNN-LSTM models have a higher number of parameters compared to the LSTM and GRU models, indicating a higher model complexity. This increased complexity allows the CNN-LSTM models to capture more intricate spatial and temporal relationships in the data.

Table 5.3: CNN LSTM model performance comparison

CNN-LSTM architecture				
Models	CNN-LSTM - lat	CNN-LSTM - lon	**CNN-LSTM - RDA(d)**	**CNN-LSTM - RDA(a)**
Number of convolutional layers	1	1	1	1
Kernel size (in convolutional layer)	3	3	3	1
Filter size (in convolutional layer)	32	32	32	32
Number of hidden layers	2	2	2	2
Size of input layer	10*8	10*8	10*8	10*8
Size of output layer	10*1	10*1	10*1	10*2
Ideal number of neurons ** ROT[2.2]	73.3	73.3	73.3	73.3
Number of neurons (LSTM 1^{st} layer)	74	74	74	74
Number of neurons (LSTM 2^{st} layer)	74	74	74	74
Number of parameters	76,726	76,726	76,726	76,214
Number of epochs	18	12	20	20
Learning rate	0.0001	0.0001	0.0001	0.0001
Lowest error epoch	20	19	20	19
Test set error (MSE)	1.96E-06	2.69E-06	0.000385	0.0388
Mean Displacement Error (nm)		0.091		0.052
Median Displacement Error (nm)		0.059		**0.021**
Final Displacement Error (nm)		0.091		**0.052**

5.4 Cumulative comparison

This section presents a comprehensive analysis of the model performance comparison, focusing on three different models: LSTM, GRU, and CNN-LSTM. Tables 5.1, 5.2, and **??** provide the detailed performance metrics for each model. Some of the keys from results are:

- The RDA approch on averge helps reduce the median error by 65%.
- GRU models carry the least amount of parameters.
- Descending Performance order for the RDA dataset: CNN LSTM > LSTM > GRU
- CNN-LSTM model demonstrates the best performance when applied to the RDA dataset, with a median displacement error of 0.021 nm (equivalent to 38 meters).

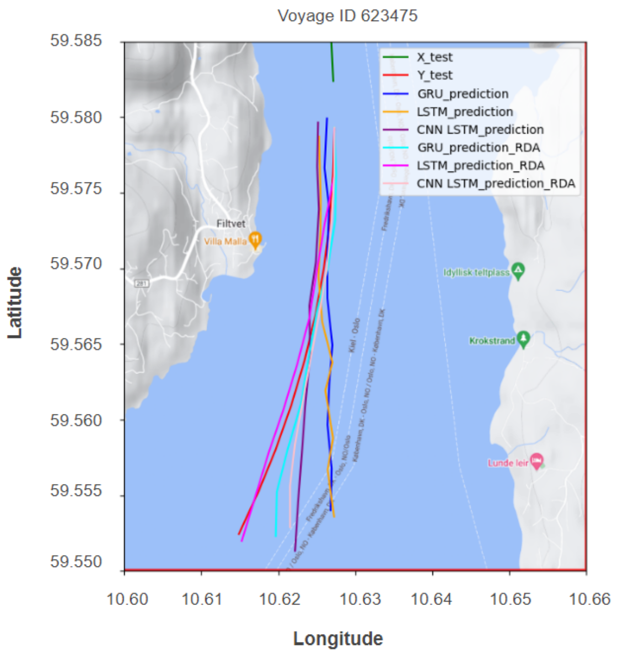


Figure 5.1: Trajectory prediction plot- Non linear trajectory

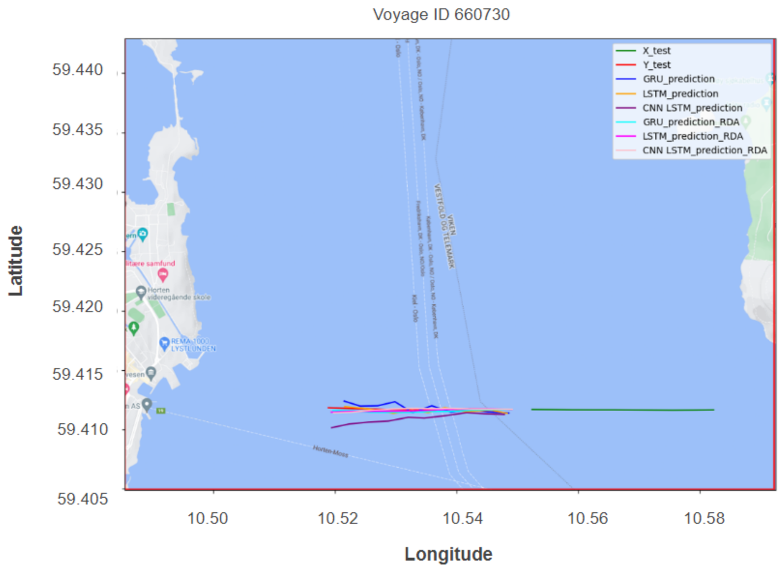


Figure 5.2: Trajectory prediction plot- Linear trajectory

Chapter 6

Discussion

The results demonstrate that the RDA approach reduces the prediction errors significantly. The median displacement error is approximately 65% lower for all models, indicating their improved performance in capturing trajectory patterns and making accurate predictions. By incorporating relative angle mapping, the models are able to capture non-linear patterns more effectively, as also observed from the trajectory plots in figures 5.1 and 5.2. This allows for a more accurate representation of ship trajectories in confined waters.

Furthermore, it is noteworthy that the GRU models exhibit a 25% lower number of parameters compared to LSTM models, making them computationally efficient while still delivering comparable accuracy. This makes GRU models a favorable choice when computational resources are limited. Among all the models, the CNN-LSTM model stands out as the best performer when applied to the RDA dataset. It achieves a median displacement error of 0.021 nm (equivalent to 38 meters), demonstrating its capability to capture spatial and temporal dependencies effectively.

Overall, the RDA approach, combined with GRU or CNN-LSTM models, proves to be advantageous in short-term trajectory prediction in confined waters. By incorporating the approach with appropriate models, the accuracy of predictions can be significantly enhanced, contributing to safer navigation and decision-making in maritime operations.

Chapter 7

Conclusion and Further work

The findings of this thesis highlight the effectiveness of the RDA approach in improving the accuracy of short-term trajectory prediction in confined waters. By reducing the vocabulary and enhancing the relation mapping, the RDA approach enables better learning of state inter-dependencies, resulting in more accurate predictions.

To ensure real-world applicability, the proposed approach was implemented and evaluated using 10 months of training data, with 1 month for validation and the final month (December'19) as the test set. This approach allows for a comprehensive analysis of the model's performance and its ability to generalize to unseen data, promoting real world repeatability.

Additionally, a multi-step framework was developed and employed for effective data processing prior to training the models. This framework takes into account the temporal dependencies in ship trajectories, enabling a more comprehensive prediction of future positions.

Moving forward, there are several avenues for further research. Some of these are:

1. Consideration encounter scenes: The current modelling approaches and datasets do not explicitly cover encounter scenes, which can have a significant impact on ship trajectories, especially in confined waters. Exploring alternative representation techniques or incorporating additional contextual information specific to encounter scenarios can improve the models' ability to predict trajectory patterns accurately in such situations.

2. Advanced modelling approaches: While the RDA approach has shown promising results when combined with standard models, further exploration of advanced modelling approaches can help enhance prediction accuracies. Techniques such as ensemble learning, hybrid models combining different architectures, or the integration of domain-specific knowledge into the models can potentially lead to improved performance.

3. Incorporation of physics-based models: The use of spline functions and hybrid modelling approaches can be explored to make the prediction process more physics-

based and realistic. By incorporating physical laws and constraints that govern ship movements, the models can better capture the underlying dynamics and interactions in confined waters, resulting in more accurate trajectory predictions.

4. Enhanced feature engineering: More comprehensive feature engineering techniques can be employed to generate spatial features and establish better correlations between different aspects of ship trajectories. This may involve considering additional input variables, such as vessel characteristics, environmental factors, or navigational constraints, and incorporating them into the models to capture a wider range of influencing factors.

5. Open source Trajectory Dataset: A significant limitation in the field of trajectory prediction is the lack of a standardized and openly accessible dataset for evaluation and comparison purposes. There is a need for an open source trajectory dataset that covers diverse scenarios and encompasses various navigation conditions in maritime settings. By creating such a dataset and making it available to the research community, researchers can ensure consistent evaluation and facilitate advancements in the field. The availability of an open source dataset would foster collaboration, enable benchmarking, and contribute to the development of more reliable and generalizable prediction models.

By pursuing these research directions, it is possible to enhance the prediction accuracy and robustness of short-term trajectory models in confined waters. These advancements would contribute to the development of more reliable and effective tools for maritime navigation, collision avoidance systems, route planning, and maritime traffic management, ultimately improving safety and efficiency in maritime operations.

Bibliography

Maurycy Pietrzak, . vincenty. URL: https://pypi.org/project/vincenty/.

Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., Savarese, S., 2016. Social lstm: Human trajectory prediction in crowded spaces, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 961–971. doi:10.1109/CVPR.2016.110.

AS, G.S., Year. Kart (map). URL: https://kart.gulesider.no/?c=59.474907,10.628586&z=11&l=nautical.

Balthazar Rouberol, . haversine. URL: https://pypi.org/project/haversine/.

Cho, K., Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H., Bengio, Y., 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation doi:10.3115/v1/D14-1179.

Christopher Olah, . Understanding lstm networks. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Dalsnes, B.R., Hexeberg, S., Flåten, A.L., Eriksen, B.O.H., Brekke, E.F., 2018. The neighbor course distribution method with gaussian mixture models for ais-based vessel trajectory prediction, in: 2018 21st International Conference on Information Fusion (FUSION), pp. 580–587. doi:10.23919/ICIF.2018.8455607.

Daoulas, V., Tampouratzis, N., Mousouliotis, P., Papaefstathiou, I., 2021. An open-source implementation of lstm and gru in the ptolemy simulation framework, in: 2021 IEEE/ACM 25th International Symposium on Distributed Simulation and Real Time Applications (DS-RT), pp. 1–8. doi:10.1109/DS-RT52167.2021.9576137.

Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L., 2009. Imagenet: A large-scale hierarchical image database, in: 2009 IEEE conference on computer vision and pattern recognition, Ieee. pp. 248–255.

European Environmental Agency, . Rail and waterborne — best for low-carbon motorised transport. URL: https://www.eea.europa.eu/publications/rail-and-waterborne-transport#:~:text=Rail%20and%20waterborne%20transport%20have,road%20transport%20emit%20significantly%20more.

European Maritime Safety Agency, . Annual overview of marine casualties and incidents 2019, technical report. (2019). URL: https://www.emsa.europa.eu/publications/reports/item/3734-annual-overview-of-marine-casualties-and-incidents-2019.html.

Fukushima, K., 1980. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics 36, 193–202.

Google, . Framing an ml problem. URL: https://developers.google.com/machine-learning/problem-framing/ml-framing.

Greff, K., Srivastava, R.K., Koutník, J., Steunebrink, B.R., Schmidhuber, J., 2017. Lstm: A search space odyssey. IEEE Transactions on Neural Networks and Learning Systems 28, 2222–2232. doi:10.1109/TNNLS.2016.2582924.

Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M.H., Brett, M., Haldane, A., del Río, J.F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., Oliphant, T.E., 2020. Array programming with NumPy. Nature 585, 357–362. URL: https://doi.org/10.1038/s41586-020-2649-2, doi:10.1038/s41586-020-2649-2.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778. doi:10.1109/CVPR.2016.90.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural computation 9, 1735–1780.

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

Imber, J., Tings, B., Velotto, D., 2019. Simultaneous estimation of multiple ship parameters from sar images using a forked convolutional neural network, in: IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium, pp. 310–313. doi:10.1109/IGARSS.2019.8898436.

IMO, . Ais transponders. URL: https://www.imo.org/en/OurWork/Safety/Pages/AIS.aspx.

Jordahl, K., den Bossche, J.V., Fleischmann, M., Wasserman, J., McBride, J., Gerard, J., Tratner, J., Perry, M., Badaracco, A.G., Farmer, C., Hjelle, G.A., Snow, A.D., Cochran, M., Gillies, S., Culbertson, L., Bartos, M., Eubank, N., maxalbert, Bilogur, A., Rey, S., Ren, C., Arribas-Bel, D., Wasser, L., Wolf, L.J., Journois, M., Wilson, J., Greenhall, A., Holdgraf, C., Filipe, Leblanc, F., 2020. geopandas/geopandas: v0.8.1. URL: https://doi.org/10.5281/zenodo.3946761, doi:10.5281/zenodo.3946761.

Kimbra Cutlip, . Ais for safety and tracking: A brief history. URL: https://globalfishingwatch.org/data/ais-brief-history/.

Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 .

Krishanth, K., Tharmarasa, R., Kirubarajan, T., Valin, P., Meger, E., 2012. Prediction, tracking, and retrodiction for path-constrained targets. Proceedings of SPIE - The International Society for Optical Engineering 8393, 18–. doi:10.1117/12.921035.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks, in: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, Curran Associates Inc., Red Hook, NY, USA. p. 1097–1105.

Lecun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, 2278–2324. doi:10.1109/5.726791.

Liu, J., Shi, G., Zhu, K., 2019. Vessel trajectory prediction model based on ais sensor data and adaptive chaos differential evolution support vector regression (acde-svr). Applied Sciences 9. URL: https://www.mdpi.com/2076-3417/9/15/2983, doi:10.3390/app9152983.

Mao, S., Tu, E., Zhang, G., Rachmawati, L., Rajabally, E., Huang, G.B., 2018. An automatic identification system (ais) database for maritime trajectory prediction and data mining, in: Cao, J., Cambria, E., Lendasse, A., Miche, Y., Vong, C.M. (Eds.), Proceedings of ELM-2016, Springer International Publishing, Cham. pp. 241–257.

Mayank Banoula, . An overview on multilayer perceptron (mlp). URL: https://www.simplilearn.com/tutorials/deep-learning-tutorial/multilayer-perceptron.

Wes McKinney, 2010. Data Structures for Statistical Computing in Python, in: Stéfan van der Walt, Jarrod Millman (Eds.), Proceedings of the 9th Python in Science Conference, pp. 56 – 61. doi:10.25080/Majora-92bf1922-00a.

Ministry of Transport, . Regulations on pilot duty and the use of pilotage certificates (pilot duty regulations). In 2014 booklet 18 p 2924. Entry into force: 01.01.2015.

NHTSA, . Road safety facts - annual global road crash statistics. URL: https://www.asirt.org/safe-travel/road-safety-facts/#:~:text=Approximately%201.3%20million%20people%20die,resulting%20in%20long%2Dterm%20disabilities.

Perera, L., Guedes Soares, C., 2010. Ocean vessel trajectory estimation and prediction based on extended kalman filter.

Rosin, F., Forget, P., Lamouri, S., Pellerin, R., 2022. Enhancing the decision-making process through industry 4.0 technologies. Sustainability 14. URL: https://www.mdpi.com/2071-1050/14/1/461, doi:10.3390/su14010461.

Sainath, T.N., Vinyals, O., Senior, A., Sak, H., 2015. Convolutional, long short-term memory, fully connected deep neural networks, in: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4580–4584. doi:10.1109/ICASSP.2015.7178838.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: A simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15, 1929–1958.

Suo, Y., Chen, W., Claramunt, C., Yang, S., 2020. A ship trajectory prediction framework based on a recurrent neural network. Sensors 20. URL: https://www.mdpi.com/1424-8220/20/18/5133, doi:10.3390/s20185133.

Tang, H., Yin, Y., Shen, H., 2019. A model for vessel trajectory prediction based on long short-term memory neural network. Journal of Marine Engineering & Technology 21, 136–145. URL: https://doi.org/10.1080/20464177.2019.1665258, doi:10.1080/20464177.2019.1665258, arXiv:https://doi.org/10.1080/20464177.2019.1665258.

Thyri, E.H., Breivik, M., Lekkas, A.M., 2020. A path-velocity decomposition approach to collision avoidance for autonomous passenger ferries in confined waters. IFAC-PapersOnLine 53, 14628–14635. URL: https://www.sciencedirect.com/science/article/pii/S240589632031884X, doi:https://doi.org/10.1016/j.ifacol.2020.12.1472. 21st IFAC World Congress.

United States Coast Guard, . Automatic identification system (ais) – accurate broadcasts don't happen automatically. URL: https://www.dco.uscg.mil/Portals/9/DCO%20Documents/5p/CG-5PC/INV/Alerts/USCGSA_0420.pdf?ver=2020-05-13-090105-050.

Van Rossum, G., 2020. The Python Library Reference, release 3.8.2. Python Software Foundation.

Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, İ., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., SciPy 1.0 Contributors, 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods 17, 261–272. doi:10.1038/s41592-019-0686-2.

Appendix

GitHub repository with code files:
https://github.com/NeerajMehta04/TMR-4930-Ship-trajectory-prediction-in-confined-waters.git