2023

Spatio-Temporal Deep Learning Approaches for Addressing Track Association Problem using Automatic Identification System (AIS) Data

Md Asif Bin Syed
West Virginia University, ms00110@mix.wvu.edu

Spatio-Temporal Deep Learning Approaches for Addressing Track Association Problem using Automatic Identification System (AIS) Data

Md Asif Bin Syed

Thesis submitted to the **Benjamin M. Statler College of Engineering and Mineral Resources** at West Virginia University
in partial fulfillment of the requirements for the degree of
MS in
Industrial Engineering

Department of Industrial and Management Systems Engineering

Morgantown, West Virginia
2023

Keywords: Track Association, Marine Surveillance, Deep Learning, 1D CNN LSTM, Graph Convolutional Neural Network, Temporal-GCN, Multiple Object Tracking, Automatic Identification System (AIS), Spatio-temporal Algorithm

Abstract

Spatio-Temporal Deep Learning Approaches for Addressing Track Association Problem using Automatic Identification System (AIS) Data

Md Asif Bin Syed

In the realm of marine surveillance, track association constitutes a pivotal yet challenging task, involving the identification and tracking of unlabelled vessel trajectories. The need for accurate data association algorithms stems from the urge to spot unusual vessel movements or threat detection. These algorithms link sequential observations containing location and motion information to specific moving objects, helping to build their real-time trajectories. These threat detection algorithms will be useful when a vessel attempts to conceal its identity. The algorithm can then identify and track the specific vessel from its incoming signal. The data for this study is sourced from the Automatic Identification System, which serves as a communication medium between neighboring ships and the control center. While traditional methods have relied on sequential tracking and physics-based models, the emergence of deep learning has significantly transformed techniques typically used in trajectory prediction, clustering, and anomaly detection. This transformation is largely attributed to the deep learning algorithm's capability to model complex nonlinear relationships while capturing both the spatial and temporal dynamics of ship movement. Capitalizing on this computational advantage, our study focuses on evaluating different deep learning architectures such as Multi Model Long Short-Term Memory(LSTM), 1D Convolutional-LSTM, and Temporal-Graph Convolutional Neural Networks— in addressing the problem of track association. The performance of these proposed models are compared against different deep learning algorithms specialized in track association tasks using several real-life AIS datasets.

Contents

List of Tables

List of Figures

List of Abbreviations and Symbols

Abbreviations

AIS	Automatic Identification System
MMSI	Maritime Mobile Service Identity
MOT	Multiple Object Tracking

CHAPTER 1

Introduction

Maritime surveillance serves as an essential component in both military and civilian sectors, functioning as a pivotal instrument for ensuring safety and maintaining security across various marine environments. A particularly salient challenge within this domain is the problem of track association. This study aims to address the track association problem by utilizing spatio-temporal algorithms applied to data generated from the Automatic Identification System (AIS). To elaborate on the track association problem and our contribution, the ensuing segments of this chapter will be systematically organized into the following sections: Track Association and Automatic Identification System, Motivation, Objective of the Study, and Data Description, thereby providing a coherent framework for understanding the problem.

1.1 Track Association and Automatic Identification System

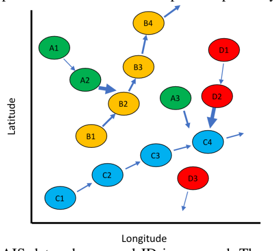
The problem of the track association involves the accurate assignment of moving entities to their respective trajectories. It helps to pinpoint and distinguish potential candidates posing threats from other normal objects and monitor the anomalous trajectories until intervention [1]. The challenge of track association can be tackled using various types of maritime data, including satellite imagery, signals emitted from nearby ships, radar data, and readings from assorted sensors. Notably, the Automatic Identification System (AIS) emerges as a focal point of scholarly attention in this domain. AIS predominantly functions on the back of self-reported data from vessels in conjunction with satellite-derived information. It is instrumental in amplifying the time-sensitive resolution, widening geographic oversight, and bolstering both near real-time and prolonged surveillance capacities [2].

Introduced by the International Maritime Organization (IMO) in the year 2000, AIS is now a ubiquitous technology, mandated for installation in commercial ships exceeding certain tonnage, as well as all passenger ships. Functionally, AIS operates through an ensemble of hardware and software components that include one VHF transmitter, two VHF receivers, a GPS receiver, a CPU with specialized software, and a display unit. This comprehensive setup allows the collection of a wide array of vessel-related information. On the one hand, there is static information, which encompasses vessel identifiers like the IMO number, Maritime Mobile Service Identity (MMSI) number, and other physical attributes such as the size and type of the vessel. On the other hand, AIS also captures dynamic voyage-related data, including geographical coordinates (latitude and longitude), speed, and heading direction. This data is then broadcasted to both maritime authorities on the coast and other vessels, enhancing real-time situational awareness.

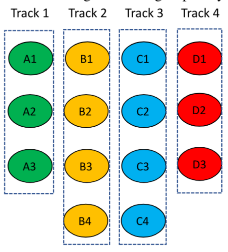
The primary function of the Automatic Identification System (AIS) was initially to prevent collisions between vessels, thereby enhancing maritime safety [3]. However, the scope of AIS has since expanded to serve as a comprehensive monitoring system for tracking vessel movements both in open seas and near ports. This expanded role is partially due to AIS's superior performance in locating vessels under conditions of limited or zero visibility compared to traditional radar systems. Such capabilities enable maritime authorities and watch guards to take proactive measures [4]. Consequently, AIS has become an invaluable tool not only for collision avoidance but also for maritime traffic management and coastal security maintenance. In terms of data, AIS furnishes kinetic information, including current geographic coordinates and dynamic infoormation such as Speed Over Ground (SOG) and Course Over Ground (COG). Additionally, AIS provides voyage details as well as static information, such as the Maritime Mobile Service Identity (MMSI) number and vessel name, facilitating a comprehensive understanding of marine activities in a given area [5]. Depending on the goal of the model, the static data can be used for different modeling techniques such as two factor regression analysis which can be utilized in trajectory prediction [6].

The Track Association Problem in the context of AIS (Automatic Identification System) data involves identifying and associating discrete data points, or "nodes," to their corresponding maritime vessels when vessel identification information is absent . In a graphical representation shown in figure1.1(a), when AIS data is complete with vessel IDs, each node can be easily tagged to a specific vessel and its trajectory (Track 1, Track 2, etc.)(see in figure 1.1(b)). However, in figure 1.1(c), when this identification information is removed, the nodes are merely represented as object IDs generated sequentially based on their timestamps. Consequently, the track association problem focuses on reconstructing these tracks by leveraging the existing spatial and temporal information such as timestamp, spatial coordinates (longitude and latitude) and other dynamic attributes like speed and direction (See in figure 1.1(d)).

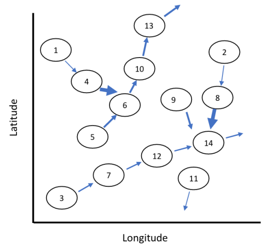
(a) Graphical depiction of the AIS data. Each letter and color code represent an unique vessel. Direction and width of the arrows represent vessel course and speed respectively.



(b) Tracks consisting of nodes grouped by vessels.



(c) The AIS data when vessel ID is removed. The numbers represent the objects IDs generated sequentially with respect to their timestamps.



(d) Track association in absence of vessel ID.

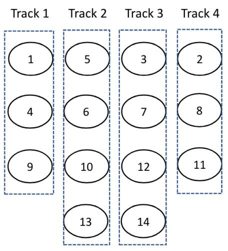


Figure 1.1: Track association problem

1.2 Motivation

A significant challenge in track association arises from occlusions or target breakages, which are prevalent in marine monitoring scenarios. Vessels engage in various maneuvers, encounter occlusions, or navigate through congested channels, all of which can lead to track interruptions or discontinuities [7]. These disruptions introduce substantial difficulties in maintaining continuous and precise track associations. In this particular scenario, MMSI number is missing from the AIS system.

The intricacies of track association extend beyond occlusions, encompassing the issue of time gaps in the vessel reporting system. Vessels can temporarily cease transmitting signals for numerous reasons, ranging from equipment malfunctions to deliberate concealment of their trajectory. Prolonged signal absences introduce significant challenges in correctly associating nodes. During such gaps, vessels can undergo abrupt changes in direction, experience drastic alterations in speed, or even come to a complete halt. These uncertainties complicate the establishment of a consistent and accurate vessel trajectory.

Furthermore, the nature of the targets being tracked adds another layer of complexity to track association [8] challenge. Notably, challenges intensify when vessels operate in proximity to ports. Ports often host a multitude of vessels in close quarters, making it arduous to differentiate between them. Additionally, vessels near ports frequently engage in maneuvers, whether for parking or creating space for other vessels. As a result, tracking individual vessels in such environments necessitates specialized attention and techniques to ensure precise and reliable track association.

The scale of marine surveillance further compounds the challenges of track association. In large-scale monitoring scenarios, a high density of vessels can lead to signal interference and difficulties in distinguishing between high-density and low-density areas [9]. These factors can have a detrimental impact on the accuracy of track associations, particularly in regions with overlapping tracks.

In situations where the MMSI number is missing or compromised, the fundamental question arises: Can one accurately associate these time-sequenced nodes within an AIS dataset to reconstruct vessel trajectories? To address track association challenges, various approaches and algorithms have been proposed. These include nearest neighbor approaches, likelihood-based algorithms, kalman filtering and adaptive filtering techniques [10, 11, 12, 13]. These methods aim to improve the accuracy and robustness of track association in marine surveillance. However, these methods are more suitable for tracking linear motion and suffer in the presence of nonlinear trajectories with multiple overlapping tracks and missing data. These issues can be better handled by the physics-based models. While tracking, they consider the spatio-temporal patterns of the vessels and proved to be more effective for tracking maritime vessels than these approaches [1]. However, these frameworks use only the most recent datapoint for track association, thereby ignoring all the previous datapoints and the information hidden in them. These sequences often contain the long-term trajectory pattern of vessels and can help in future track association. Deep learning architectures are proven to be quite effective in modeling nonlinear sequential data and thus can also be utilized for trajectory modeling. Recently, deep learning algorithms have been proposed [14] to automatically learn hierarchical feature representations using raw spatio-temporal data quite similar to our case. Therefore, deep learning methods offer a promising solution for understanding intricate patterns essential for reliable AIS-driven marine vessel tracking. This study aims to explore different deep learning architectures to enable accurate trajectory reconstruction and track association from AIS datasets.

1.3 Objective of the Study

The objective of this work can be summarized as follows:

- ☑ To propose spatio-temporal track association algorithms that can associate vessel measurements collected from the AIS to their true tracks in real-time.
- ☑ To design a multimodal Long Short-Term Memory (LSTM) framework to model the trajectory of vessels.
- ☑ To introduce and evaluate a novel 1D Convolutional-LSTM architecture, tailored for AIS datasets and the track association task.
- ☑ To design and construct a Temporal Graph Convolutional Neural Network (T-GCNN) architecture, wherein each data point is interconnected through location-based edges, augmented by a temporal layer such as LSTM or GRU to capture sequential features.
- ☑ To compare the performance of the different neural net architectures and discuss their relative advantages and disadvantages.

1.4 Data Description

This section provides an overview of the datasets of the Automatic Identification System (AIS) that will be utilized for the development and testing of our proposed framework. The AIS dataset was obtained from the National Automatic Identification System, which is managed by the Coast Guard department of a particular country. Upon reception of a signal within the Automatic Identification System (AIS) framework, the incoming data is allocated a distinct object identifier. Subsequently, a monitoring officer undertakes the task of verifying if the vessel identifier is already recorded within the preexisting database. The vessel identification (VID), including alphanumeric characters, functions as a distinct identifier for individual vessels inside the system.

ID	VID	SEQUENCE_DTTM	LAT	LON	SPEED	COURSE
1	vessel 1	2020-02-29T22:00:01Z	37.8567167	23.53735	0	0
2	vessel 2	2020-02-29T22:00:01Z	37.9483	23.6410167	0	349.9
3	vessel 3	2020-02-29T22:00:01Z	37.9390233	23.6688483	0	228.3
4	vessel 4	2020-02-29T22:00:01Z	37.93884	23.6686333	0	0.1
5	vessel 5	2020-02-29T22:00:02Z	37.9314717	23.6804267	0	170.1
6	vessel 6	2020-02-29T22:00:02Z	37.9131117	23.5476617	0.1	33.3

Figure 1.2: Automatic Identificaition System(AIS) dataset from a specific region

If the information that is received corresponds to a VID that has been previously recorded, it is stored using the same identifier. In the event that the vessel's identification cannot be ascertained, it is allocated an identifier that is derived from the timestamp. The AIS data encompasses necessary details such as velocity, the direction of motion, and geographical coordinates, denoted by longitude and latitude. The aforementioned timestamps function as nodes in the context of the track association problem.

The database used in this study was set up as a CSV file. A subset of 327 vessels is selected from the database where the time intervals between signals are inconsistent. Traditional track association algorithms are usually challenged by these uneven signal intervals. Approximately 25,000 data points are used for training and 5,000 vessel IDs are set aside for testing in order to compare existing algorithms.

Seven parameters make up the dataset, including object ID, which serves as the database's primary key. As illustrated in Figure 1.2, the other characteristics include vessel ID, timestamp (which includes date and time), latitude and longitude stated in degrees, speed expressed in knots, and course expressed in degrees. Labels like "vessel 1," "vessel 2," and so forth have taken the place of long string combinations of alphanumeric characters that are exclusive to each vessel in order to improve clarity.

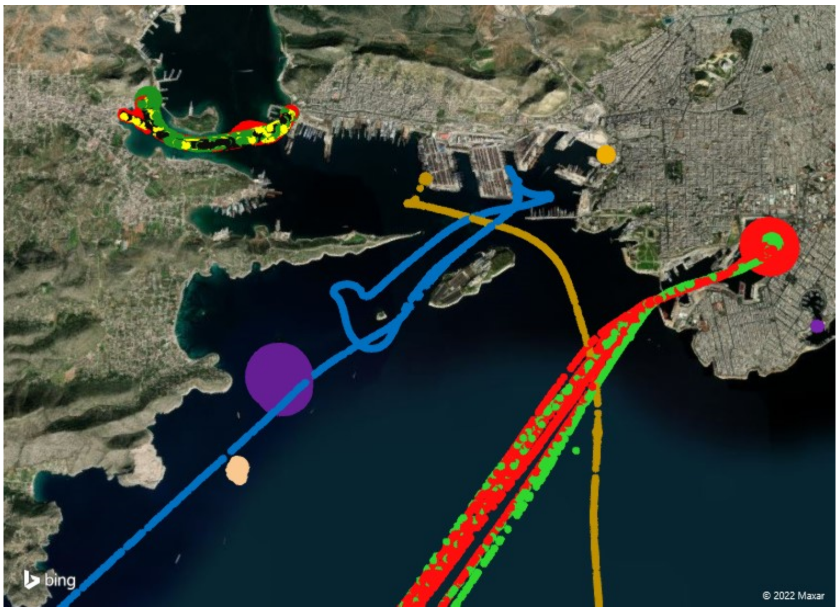


Figure 1.3: Visualization of vessels and associated tracks in the AIS dataset. Each color corresponds to a track from a different vessel. The map demonstrates the overlapping of the tracks of vessels which makes the track association problem challenging

Timestamps, geographic data in the form of longitude and latitude, and dynamic factors like speed over ground and course of direction are chosen as the main input parameters for the model design. These chosen parameters take both spatial and temporal information into consideration. The target variable is set to be the vessel ID. The complexity of the track association problem we intend to study is illustrated visually in figure 1.3. The chosen vessels have overlapping tracks, which creates a difficult scenario that mimics the complexity found in real-world circumstances. The figure's colors each represent a distinct Vessel ID.

The structure of the remaining chapters is organized as follows: Chapter 2 delves into a comprehensive literature review. Chapter 3 elucidates the methodology with a detailed research design. The comparative analysis of the results generated from multi-modal LSTM, 1D CNN-LSTM and temporal GCN is presented in Section 4 and finally, the thesis is concluded in Section 5.

CHAPTER 2

Literature Review

The study of AIS data has been ongoing for decades, with researchers exploring various facets of this data. The researchers employed various literature review techniques, such as systematic and bibliometric literature reviews [15] to obtain a broad understanding of the research topic. Several types of studies, including trajectory prediction, trajectory classification, anomaly detection, and the track association problem, have harnessed the capabilities of Automatic Identification System (AIS). In this chapter, we embark on a comprehensive review of the existing body of knowledge concerning track association techniques, with a specific focus on the application of spatio-temporal algorithms to data derived from the AIS. To ensure a systematic and coherent exploration, this chapter is thoughtfully organized into distinct segments, each dedicated to a specific aspect of track association techniques. These segments encompass Multiple Object Tracking, Sequential Tracking Models, Physics-Based Models, and Neural Network-Based Models.

2.1 Multiple Object Tracking (MOT)

The process of monitoring an object over a sequence of frames to determine its location and direction is referred to as object tracking [16]. Both single and multi-object tracking techniques have been used in various applications, including computer vision problems, human behavior analysis, and security surveillance to name a few. Multiple Object Tracking (MOT) can be particularly useful in the context of marine security and surveillance, as they can combine information from multiple sensors like radar and sonar, instead of relying solely on AIS data and track several vessels in real-time [1].

However, applying MOT approaches to AIS data for track association is not straightforward and presents several challenging issues. Firstly, these approaches must appropriately handle the ambiguity surrounding the number of vessels. Sometimes this information is unknown and must be learned on the go. Secondly, object (vessel) appearance and disappearance are often uncertain, as they may emerge at unexpected times and locations and depart at the tracking boundary. Finally, multiple incidences of overlapping tracks and time gaps must be considered during the tracking process [1]. Over the past several decades, numerous research efforts have been undertaken to develop innovative and efficient solutions to multiple object-tracking problems. However, very few of them can handle all the issues posed by the AIS track association problem. The following subsections summarize these efforts into a few major areas namely sequential tracking models, physics-based models, machine learning-based and hybrid models, and finally deep learning approaches.

2.1.1 Sequential Tracking Models

Sequential tracking algorithms, such as global nearest neighbor (GNN) [17] and joint probabilistic data association (JPDA) [18], are commonly employed to update tracks based on the contribution of several objects. These algorithms utilize a cost assignment matrix to minimize costs and employ soft assignment, also known as track association probability, to achieve this goal. While Global Nearest Neighbor (GNN) [17] and Joint Probabilistic Data Association (JPDA) [18] focus on a single hypothesis for tracking objects, there are other techniques available. For example, multiple hypotheses tracking (MHT) [19] constructs a tree of hypotheses for each item and computes the likelihood of each track to determine the most probable combination of tracks. Random finite set (RFS) based approaches have also been utilized for tracking objects, as they are capable of handling the inherent uncertainty involved in the tracking process [20]. The majority of Sequential tracking-based algorithms are based on the Kalman filtering (KF) approach [21] or its variations, which are frequently employed to track moving objects and provide information regarding their velocity and acceleration based on their position. However, the accuracy of KF is predicated on the assumption of linear motion, and it struggles to accommodate non-linear motion patterns. Furthermore, the KF framework has limited capacity for handling the distinct characteristics of vessel movements.

2.1.2 Physics-based Models

The physics-based approaches rely on mathematical equations to describe the motion of ships, taking into account factors such as mass, force, and inertia. These equations utilize physical laws to calculate the future motion characteristics of the ship [22, 23, 24, 25]. Such motion models can be useful for developing simulation systems to study ideal ship kinematic characteristics or even to train navigation systems. However, applying these models to track the trajectory patterns of multiple ships can be challenging. While these models can incorporate the spatiotemporal patterns of vessel movements [1] in the learning process, they are still limited in nature, only considering the last known position to track vessels.

2.1.3 Machine Learning and Hybrid Models

These methods rely solely on historical data and employ machine learning techniques to learn from past information, enabling them to predict future positions when provided with a new feature vector. These prominent machine learning methods used in trajectory prediction studies include the Gaussian process, support vector machine, principal component analysis (PCA), etc. While these methods [26, 27, 28, 29, 30] typically perform well in predicting immediate future positions, their prediction accuracy tends to decrease as the prediction time span increases. Furthermore, the performance of these models is highly dependent on the proper tuning of hyperparameters, which can be difficult to achieve. Additionally, they are not capable of processing long sequences and unraveling the spatial and temporal dependencies present in sequential observations. Hybrid approaches, on the other hand, combine physics-based models and machine learning models or different machine learning models to enhance the quality of the trajectory tracking process [31, 32, 33, 34, 35, 36]. These approaches, however, are not free from the limitations imposed by the physics-based and machine learning models.

2.1.4 Deep Learning-based Models

Deep learning, which is a subclass of machine learning models, stands out from the rest due to its superior learning capabilities. In the context of marine vessel trajectories, neural networks have been widely used for their ability to process large datasets and discover long-term patterns hidden in vessel trajectories. Because of the robust adaptability, the earliest form of neural network such as Multi-Layer Perceptron (MLP) [37] and Artificial Neural Network (ANN) [38, 39] have played a significant role in traffic and marine vessel trajectory prediction. Nevertheless, despite their wide applications, these neural networks exhibit low interpretability. Additionally, they present substantial challenges in terms of spatial and temporal information processing capability [40] since these networks are not equipped to handle such characteristics.

The exploration of incorporating sequential temporal patterns into marine ship trajectory prediction has motivated researchers to investigate the potential application of Recurrent Neural Networks (RNNs) [21]. However, RNNs encounter challenges in capturing long-term dependencies within a sequence due to the issue of vanishing gradients during backpropagation [41]. As a result, the limited long-term memory of these networks can hinder their performance when the data contains significant long-term dependencies [42, 43, 44]. Two prominent variations of Recurrent Neural Networks (RNNs), specifically Long Short-Term Memory (LSTM) [45, 46] and Gated Recurrent Unit (GRU) [47], have garnered substantial attention for their remarkable ability to uncover underlying patterns within extended input sequences, proving particularly advantageous for trajectory prediction.

Further advancements in research have led to the utilization of more efficient variants of LSTM and GRU, such as Bidirectional LSTM (Bi-LSTM) [48, 49], Bidirectional GRU (Bi-GRU) [50], Context-Aware LSTM (C-LSTM) model [51], and Multi-step Prediction LSTM (MP-LSTM) [52]. Distinct from traditional LSTM, Bi-LSTM has the ability to process data from both past and future contexts. This bidirectional information processing, encompassing both forward and backward information, empowers Bi-LSTM to capture a comprehensive understanding of the sequence. Consequently, numerous innovative models based on Bi-LSTM have been proposed for ship trajectory prediction. However, these models often exhibit significant computational complexity and limited generalization capabilities. The design parameters of these neural network-based frameworks are adjusted in real-time as the vessel progresses, enabling them to identify all potential trajectories a vessel may follow and reconstruct (predict) its trajectories for future time points [39]. Additionally, LSTM networks have demonstrated remarkable multitasking performance [53].

Another neural network, The Convolutional Neural Network (CNN), originally devised to address computer vision problems, has also been explored for trajectory prediction and classification of tracks [53] as it can help capturing the spatial patterns exist in the trajectory data. Instead of using the original features, several methods advocate the use of latent features derived from the neural network architecture. These methods leverage latent space representation using Variational Recurrent AutoEncoder (VRAE) [54] or LSTM [55]. These latent features can capture the spatial patterns present in the data. Temporal ordering and attention maps are also proven to be effective for object tracking [56].

In addition to conventional deep learning approaches, the research field has expanded to include the application of hybrid deep learning architectures directly to raw datasets. This advancement goes beyond transforming data into a latent space and aims to reveal both temporal and spatial relationships among features. It has proven to be particularly effective in extracting spatiotemporal relationships within the AIS dataset. Hybrid deep learning-based models for ship trajectory prediction, such as the integration of Bidirectional LSTM and RNN (BLSTM-RNN) [57] and CNN-LSTM-SE [58], have emerged as notable techniques due to their rapid learning and adaptability capabilities. These approaches excel in producing highly accurate results when dealing with complex and dynamic trajectory data.

However, it is important to note that these methods primarily focus on predicting the next points by considering the sequence of vessel nodes. This differs from track association, which aims to link vessels to their respective tracks. Furthermore, following the prediction route would require a separate prediction model for each vessel, which can complicate the tracking process when dealing with more than ten vessels. To address these challenges and fill the gap in the literature, in this work, we propose three different Deep learning architectures which are capable of processing the multivariate sequential data and capture the spatio temporal behaviours of the data for accurately assigning them to their correct track. These architectures are compared with the standalone Convolutional Neural Network (CNN), standalone Long short term memory (LSTM), Bidirectional-LSTM(Bi-LSTM) and Bidirectional-Gated Recurrent Unit (Bi-GRU).

CHAPTER 3

Methodology

The methodology chapter is structured into four core sections. It begins with data preprocessing, which is the process of cleaning and preparing the data for ingestion into the proposed models. Section 3.2 explains the foundational deep learning models to understand the architectures of the proposed models. The proposed models are discussed in section 3.3. The chapter concludes in section 3.4 with a discussion of performance metrics that have been used to evaluate the proposed models with other deep learning architectures.

3.1 Data Preprocessing

The AIS data was collected and stored in a large CSV file, featuring data from 327 vessels. The preprocessing procedure begins by splitting the vessels into batches, with each batch comprising approximately 30 vessels. Subsequently, each batch undergoes a preprocessing step, encompassing data cleansing operations. Any timestamp that contains a null value is removed during this stage. Following completion of the cleaning process, it has been observed that a considerable proportion of vessels lack sufficient data to be chosen for training our model.

Model training parameters highly depend on the number of data points available for training. To ensure good accuracy, a threshold of 50 observations (for each vessel) is set to ensure that the model receives sufficient data for training. The data distribution for batch 1 is illustrated in Figure 3.1. Initially, before the preprocessing, it is supposed to contain 30 vessels for training. However, 23 vessels from this batch are eventually used as 7 of them do not meet the data point threshold. In total, 63 vessels out of 327 vessels are excluded from further analysis and we proceed with the remaining 264 vessels, divided into 11 batches, for the model training.

The batch data are then normalized using the standard scaler equation provided below:

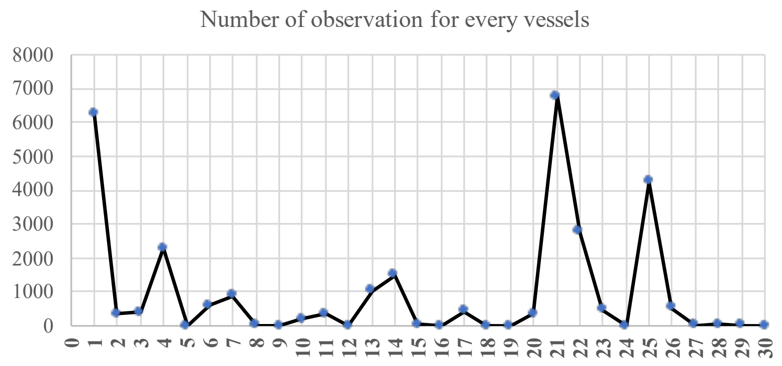


Figure 3.1: Observation frequency for each of thirty vessels

$$z = (X - u)/s \tag{3.1.1}$$

Here, X represents the input data, u is mean and s represents the unit variance.

3.2 Deep Learning Architectures

Subsequent to the data preprocessing phase, the cleaned and prepared data is ingested into the pre-designed architectural frameworks. To fortify the reader's comprehension of the intricacies of our proposed models, foundational discussions on basic neural network architectures like Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM), Gated Recurrent Units (GRU), Graph Neural Network and Graph Convolutional Neural Network are incorporated in this section. These foundational explorations serve to contextualize and substantiate the reasoning behind the selection and adaptation of the advanced architectures used in this study.

3.2.1 Convolutional Neural Network (CNN)

Convolutional neural networks (CNNs) are a type of neural network that is designed to process and analyze data with spatial relationships. CNN has been widely utilized in various applications, such as audio signal pattern recognition, image processing, natural language processing, and time series prediction. The CNN architecture proposed by Lecun et al. [59] consists of an input layer, an output layer, and multiple hidden layers. These hidden layers contain convolutional layers, which perform dot products between the input matrix and the convolution kernel [60].

Convolutional Neural Network(CNN) is particularly effective at extracting spatial patterns by using convolutional layers, which apply a set of learnable filters to the input data to detect features at different spatial locations. The convolution layer can be multi-dimensional, with its width t and height I representing the filter dimension. The output of the t-th filter, which operates on the input matrix X, is calculated below

$$k_{\mathrm{t}} = \tanh\left(a_{\mathrm{t}}.X + b_{\mathrm{t}}\right) \tag{3.2.1}$$

where X is the input matrix, b_{t} is the bias matrix, a_{t} represents the weight matrix and k_{t} is the output function. The output of the filter is then fed into the LSTM layer which is described in the following subsection.

3.2.2 Long Short Term Memory (LSTM)

Long short-term memory (LSTM) is a variant of Recurrent Neural Network (RNN) [61]. RNNs are a type of artificial neural network that can process sequential data by maintaining a memory of past inputs. In contrast to regular feedforward neural networks that process data in a single pass, RNNs are designed to handle data that has temporal dependencies, such as time series or sequences of text. One of the challenges faced by conventional RNNs is the vanishing gradient problem, in which the gradient of the error function can either diminish or explode during backpropagation when it is propagated through multiple time steps.

However, the LSTM architecture overcomes this issue by incorporating a novel memory cell that regulates the information flow through the network. This mechanism selectively retains or discards information, which mitigates the vanishing or exploding gradient problem and allows learning long-term dependencies in sequential data. LSTM enables RNNs to perform long-term sequential prediction [62]. To enable sequential learning, LSTM employs three gates: a forget gate f_t, an input gate i_t, and an output gate o_t, which are depicted in Figure 3.7. These gates control the flow of information through the memory cell. Elaborating on these three gates, the complex mechanism of LSTM architecture is explained below.

In LSTM, the forget gate plays a crucial role in determining which pieces of information to retain or discard from its existing memory, taking into account the arrival of fresh input. Let us denote the input time series as $X = (x_1, x_2, \ldots, x_t)$ and the hidden state of the memory cell as $H = (h_1, h_2, \ldots, h_{\mathrm{t}})$. The forget gate takes the concatenation of previous hidden state h_{t-1} and the current input x_t as inputs, and produces a forget vector f_t as output as in Equation 3.2.2 [63]:

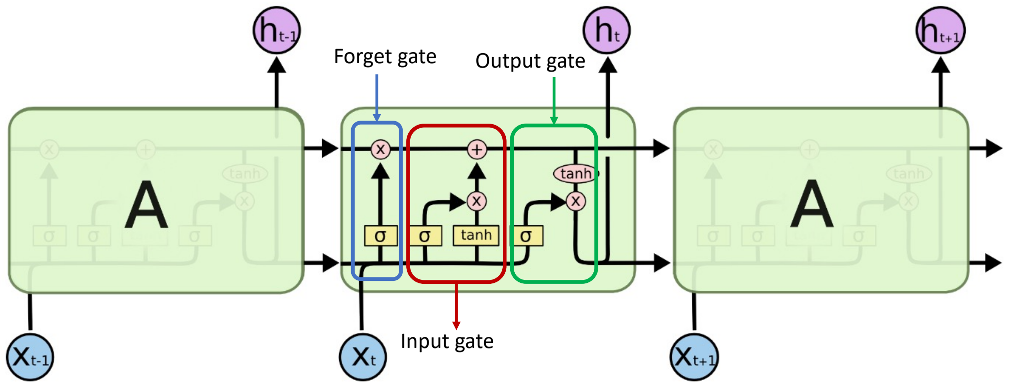


Figure 3.2: Long Short Term Memory model schematic diagram

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{3.2.2}$$

Here, σ denotes the sigmoid activation function, which is a nonlinear function that maps its input to a value between 0 and 1. The weight matrix W_f represents the weights associated with the forget gate and determines how strongly the inputs affect the forget vector. b_f represents the bias vector, which contains constant values that are added to the weighted sum of these inputs and can be thought of as the intercept term in the forget gate equation. The weights and the bias values are learned during the LSTM training process.

LSTM employs the input gate to regulate the inflow of fresh data into the memory cell, which is composed of two components: the input activation gate and the candidate memory cell gate. The input activation gate dictates the extent to which the new input ought to be incorporated into the memory cell, whereas the candidate memory cell gate governs the portion of new data that must be retained in the memory cell.

By taking the previous hidden state h_{t-1} and the current node x_t as inputs, the input gate generates an input vector i_t and a candidate memory cell vector $\tilde{c}_t$ in an LSTM. The operation of the input activation gate can be expressed using Equation (3.2.3), which involves the weight matrix W_i and bias vector b_i. Meanwhile, Equation (3.2.4) illustrates how the candidate memory cell $\tilde{c}_t$ is formed by applying the hyperbolic tangent activation function ($tanh$) to the same set of inputs, using the weight matrix W_c and bias vector b_c.

$$i_t = \sigma\left(W_i\left[h_{t-1}, x_t\right] + b_i\right) \tag{3.2.3}$$

$$\tilde{c}_t = \tanh\left(W_c\left[h_{t-1}, x_t\right] + b_c\right) \tag{3.2.4}$$

The input vector and the candidate memory cell vector are then combined to update the previous memory cell c_{t-1} (see Equation (3.2.5)). Here, $\odot$ denotes element-wise multiplication.

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{3.2.5}$$

The final gate, the output gate, controls the flow of information from the current memory cell to the current hidden state, which is also the output of the LSTM at the current time step. First, the output vector o_t is generated as in Equation (3.2.6).

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t, c_t\right] + b_o\right) \tag{3.2.6}$$

Then the current hidden state, h_t is obtained using Equations (3.2.7) and (3.2.8) where $\tilde{h}_t$ represents the candidate hidden state value for the current time step of the LSTM network.

$$\tilde{h}_t = \tanh\left(c_t\right) \tag{3.2.7}$$

$$h_t = o_t \odot \tilde{h}_t \tag{3.2.8}$$

3.2.3 Gated Recurrent Unit (GRU)

The Gated Recurrent Unit (GRU) is another variant of the standard Recurrent Neural Network (RNN) architecture. Designed to address the vanishing gradient problem commonly encountered in traditional RNNs, GRU maintains the capability to remember long-term dependencies in a sequence. Comprising fewer parameters than the Long Short-Term Memory (LSTM) model, the GRU is computationally more efficient, yet still capable of capturing complex temporal relationships.

The specific mathematical formulation of the GRU is detailed below:

The reset gate r_t is computed as in Equation 3.2.9. This gate enables the decision-making mechanism on how to amalgamate the new input x_t with the previous memory h_{t-1}.

$$r_t = \sigma\left(W_t + \left[h_{t-1}, x_t\right] + b_t\right) \tag{3.2.9}$$

Following this, the update gate z_t is formulated as in Equation 3.2.10. This gate regulates the

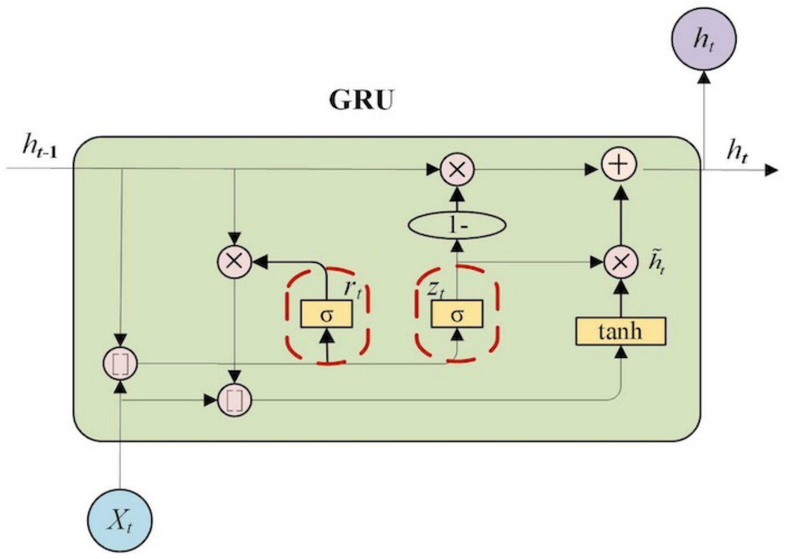


Figure 3.3: Gated Recurrent Units (GRU) schematic diagram

extent to which the hidden state is updated during the learning process.

$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t] + b_z\right) \tag{3.2.10}$$

Upon calculating the reset and update gates, we obtain the candidate hidden state $\tilde{h}_\mathrm{f}$ using Equation 3.2.11. This candidate will contribute to the formation of the final hidden state h_t.

$$\tilde{h}_\mathrm{f} = \tanh\left(W_{h_t} \cdot [r_t \Theta h_{t-1}, x_t] + b_h\right) \tag{3.2.11}$$

Lastly, the new hidden state h_t is calculated using Equation 3.2.12.

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{3.2.12}$$

In these equations, $[\cdot]$ symbolizes the concatenation of two vectors, $\odot$ indicates element-wise multiplication, Θ signifies that each element in the matrix is multiplied accordingly, and σ is the sigmoid activation function. The matrices W and vectors b are learnable parameters in the GRU architecture.

In summary, the GRU simplifies the LSTM architecture by merging certain gates and states (see in figure 3.3, thereby reducing the number of trainable parameters while retaining the capacity for modeling complex sequence dependencies.

3.2.4 Graph Neural Network

Among the various types of neural networks, Graph Neural Networks (GNNs) have gained considerable attention for their ability to handle graph-structured data. Formally, a GNN is represented as $G = (V, E)$, where V constitutes the set of vertices or nodes, and E represents the set of edges in the graph G. Within this framework, information is systematically embedded into both nodes and edges to facilitate nuanced inferences. Each node in the graph is denoted by $h_i \in V$, while each edge is represented as e_{ij}, with i and j serving as the indices for the respective nodes. Importantly, the edges in a GNN can be either directed or undirected, depending on the nature of the relationships they represent.

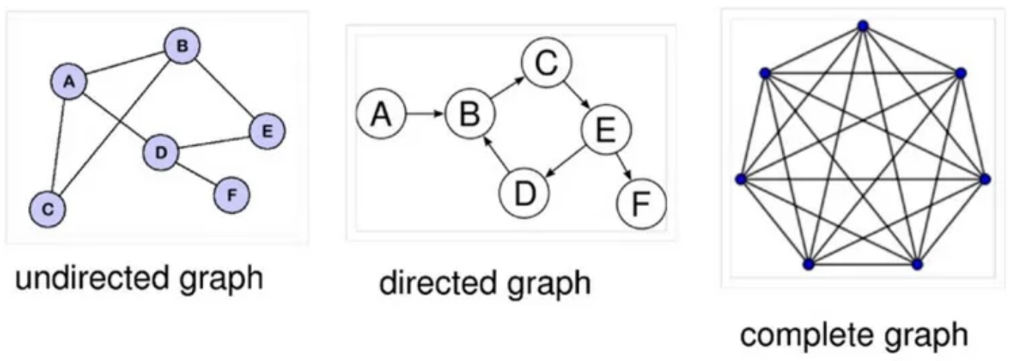


Figure 3.4: Different types of Graph Structure

In graph theory, A directed graph is a graph with all edges directed from one node to another. An undirected graph is considered a specific subset of directed graphs in which each pair of connected nodes is linked by a pair of edges with inverse directions. A graph is undirected if and only if the adjacency matrix is symmetric. While ship trajectory information itself does not have a graph structure, it contains temporal and spatial features, such as location features, distance features, and speed features. It can extract the connection relationship information between track points to establish a topological association network, which can effectively extract spatial features for machine learning. Information is encoded into the nodes and edges in these temporal graphs. Node encodings correspond to $enc\left(h_i^l\right)$ where l is the layer index and i is the node index in the corresponding graph layer. Edge encodings correspond to $\text{enc}\left(e_{ij}^l\right)$ where l is the layer index and i and j denote the corresponding node indices. Node-level, edge-level and graph-level inferences are possible by taking into account node and edge encodings [64]. In this particular study, we adopt a node-level prediction method to associate the track. Node-level outputs relate to node regression and node classification tasks. A variety of Graph neural network has been developed over the decade. Among them, GCNNs can extract high-level node representations by graph convolution. With a multi-perceptron or a softmax layer as the output layer, GCNNs are able to perform node-level tasks in an end-to-end manner [65]. The next subsection discusses the Graph Convolutional Neural Network in detail.

3.2.5 Graph Convolutional Neural Network

The Graph Convolutional Network (GCN) is a pioneering architecture in the realm of graph-based machine learning, designed to operate directly on graphs. Unlike traditional neural networks that require a fixed-size vector as input, GCNs are adept at handling irregular structures by leveraging the graph's topology. In a GCN, each node is associated with a feature vector, and the network learns to update these features based on the features of neighboring nodes. This is typically achieved through a series of layers where each layer is a non-linear function of the previous one, often involving a combination of convolutional operations on the graph and point-wise non-linearities. The adjacency matrix of the graph plays a crucial role in these operations, guiding the feature aggregation process. By doing so, GCNs are capable of capturing both local and global graph structures, making them highly effective for various tasks such as node classification, link prediction, and graph clustering.

Similar to Convolutional Neural Networks (CNNs) or Multi-Layer Perceptrons (MLPs), Graph Convolutional Networks (GCNs) learn to generate new feature representations for each node across multiple layers. These new feature vectors are then employed for linear classification tasks. For the k-th layer of the GCN, we use $\mathbf{H}^{(k-1)}$ to denote the input feature matrix for all nodes and $\mathbf{H}^{(k)}$ to represent the output feature matrix. Initially, the node features are simply the original input features, as represented by the equation:

$$\mathbf{H}^{(0)} = \mathbf{X} \tag{3.2.13}$$

This equation signifies that the original feature matrix $\mathbf{X}$ serves as the initial input for the first layer of the GCN.

In a K-layer Graph Convolutional Network (GCN), the network operation closely resembles that of a K-layer Multi-Layer Perceptron (MLP) applied to each node's feature vector $\mathbf{x}_i$. The key difference lies in the initial step of each layer, where the hidden feature representation of each node is an average of its own features and those of its neighbors. Each layer of a GCN comprises three main phases: feature propagation, linear transformation, and pointwise nonlinear activation. To facilitate comprehension, these phases are elaborated below:[66].

What sets GCNs apart from MLPs is the feature propagation stage. At the start of each layer, the feature h_i of a node v_i is averaged with the features of its surrounding nodes, as described in equation 3.2.14.

$$\tilde{\mathbf{h}}_i^{(k)} = \frac{1}{d_i+1}\mathbf{h}_i^{(k-1)} + \sum_{j=1}^{\infty} \frac{a_{ij}}{\sqrt{(d_i+1)(d_j+1)}}\mathbf{h}_j^{(k-1)} \tag{3.2.14}$$

To describe this process in a more compact form, we use the matrix operation in equation 3.2.15. Let $\mathbf{S}$ be the 'normalized' adjacency matrix that includes self-loops,

$$\mathbf{S} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}^{-\frac{1}{2}} \tag{3.2.15}$$

where $\overline{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\overline{\mathbf{D}}$ is the degree matrix of $\overline{\mathbf{A}}$. The simultaneous update in equation 3.2.14 for all nodes becomes a simple sparse matrix multiplication in equation 3.2.16

$$\hat{\mathbf{H}}^{(k)} = \mathbf{S}\mathbf{H}^{(k-1)} \tag{3.2.16}$$

This operation effectively smoothens the features at the local level and fosters similar predictions for nodes that are closely connected.

Following feature propagation, the GCN enters the stage of feature transformation and nonlinear transition. At this point, the layer in a GCN resembles a conventional layer in an MLP. Each layer is affiliated with a weight matrix $\Theta^{(k)}$ that has been learned during the training process. Subsequently, the smoothed feature vectors undergo a linear transformation. Afterward, a pointwise nonlinear activation function, typically ReLU, is employed to generate the output feature representation $\mathbf{H}^{(k)}$. Formally, the update rule for the representation of the k-th layer is defined as:

$$\mathbf{H}^{(k)} = \mathrm{ReLU}\left(\mathbf{H}^{(k)}\Theta^{(k)}\right) \tag{3.2.17}$$

The pointwise nonlinear transformation of the k-th layer is followed by the feature propagation of the $(k+1)$-th layer.

In the context of node classification tasks, the GCN behaves analogously to a standard Multilayer Perceptron (MLP) in its final layer, utilizing a softmax classifier to make predictions. Let $\hat{\mathbf{Y}} \in \mathbb{R}^{n\times C}$ represent the matrix of class probabilities for n nodes. Here, $\hat{y}_{ic}$ signifies the likelihood that node i belongs to class c. The class predictions $\hat{\mathbf{Y}}$ for a K-layer GCN can be formalized as follows:

$$\hat{\mathbf{Y}}_{\mathrm{GCN}} = \mathrm{softmax}\left(\mathbf{S}\mathbf{H}^{(K-1)}\Theta^{(k)}\right) \tag{3.2.18}$$

In equation 3.2.18, the softmax function serves as a normalizer across all classes and is defined as

$$\text{softmax}(\mathbf{x}) = \frac{\exp(\mathbf{x})}{\sum_{c=1}^{C}\exp(x_c)} \tag{3.2.19}$$

.

3.3 Proposed Deep Learning Architectures

In the section, we delve into the architecture of the proposed models, namely Multi-modal LSTM, 1D CNN-LSTM, and Temporal-GCN. The steps for implementing the architechtures have also been discussed.

3.3.1 Multi Model LSTM

The initial architecture to be implemented on the AIS dataset consists of the construction of individual models for each vessel. The predictive accuracy of these models will be assessed using the Haversine distance metric. The methodology for applying a Multimodel Long Short-Term Memory (LSTM) framework is delineated as follows:

Step 1: To train the LSTM models the available AIS messages are partitioned into training and testing sets. Let w be the number of testing data. Then, from the original dataset $X = \{x_1, x_2, \ldots, x_n\}$ of size $n \times k$, the training sequences $\{x_1, x_2, \ldots, x_{n-w}\}$ and $\{y_1, y_2, \ldots, y_{n-w}\}$ are created. Here, $x_t \in \mathbb{R}^{1\times k}$ is the input sequence, and $y_t \in \mathbb{R}$ is the output data at time t. k and n represent the number of features and the total number of observations, respectively.

Step 2: To adapt to the dimensional requirements of the LSTM architecture, the input sequence x_t is formulated by taking m continuous sequences, defined as $x_t : x_{t+m-1}$. This results in a matrix of shape $m \times k$ for $t \in \{1, 2, \ldots, n-m-1\}$.

Step 3: When the model is initiated from $t = 1$, the input layer accepts m number of timesteps as inputs and predicts the output y_{m+1}. The accuracy and validation loss of the output are subsequently calculated and integrated into the model. In these architectures, the output of one timestep serves as an input for the next timestep. This feedback mechanism empowers the model to make decisions based on the most recent input and output data. Consequently, after the prediction of y_{m+1}, it is reintroduced as an input along with the last m timesteps from the current timestep to predict the output y_{m+2}.

Step 4: This cycle is repeated for all the training data. A training stage pipeline was designed

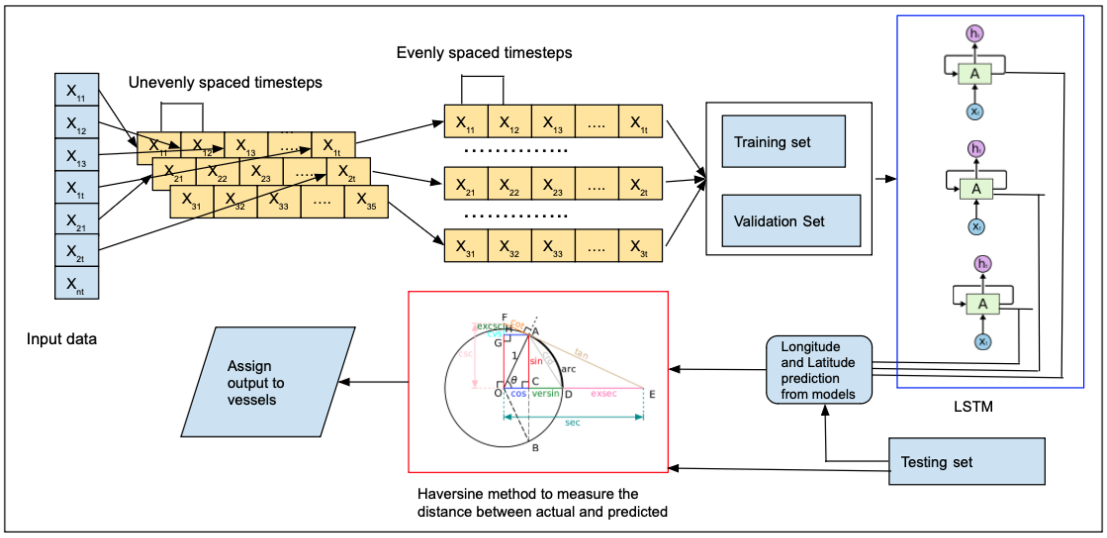

Evenly spaced timesteps
Unevenly spaced timesteps
Input data
Training set
Validation Set
Assign output to vessels
Haversine method to measure the distance between actual and predicted
Longitude and Latitude prediction from models
LSTM
Testing set

Figure 3.5: Detailed methodological framework for our proposed multi-model LSTM algorithm

to have multiple parallel input sequences for multivariate values as input, instead of flat input structures for multi-output prediction. Four features will be used which are longitude, latitude, speed and direction of the vessel for generating the multivariate input sequence.Since the vessel's position needs to be predicted, the longitude and latitude values will be generated as output.

Step 5: Once we estimate the ship's position at the times of new AIS messages, we need to find a suitable formula to calculate the similarity between the actual location and predicted location. One highly precise geodesic approach for computing the distance between two points on a sphere using their respective latitude and longitude is the haversine formula (illustrated in the red outlined box in the Figure 4.2). This formula, which is a modified version of the spherical law of cosines, is particularly well-suited for calculating small angles and distances due to its use of haversines [67]. By calculating the Haversine distance using the Equation 3.3.1, the deviation between the predicted location and actual location of the current node can be determined.

$$\text{Cdist}\left(p_k, p_n j\right) = 2r \times \arcsin\sqrt{\sin^2\left(\frac{\varphi_k - \varphi_n j}{2}\right) + \cos\varphi_k \cos\varphi_n \sin^2\left(\frac{\lambda_k - \lambda_n j}{2}\right)} \tag{3.3.1}$$

where, $\mathrm{p_k} := (\varphi_k, \lambda_k)$ and $\mathrm{p}_n\mathrm{j} := (\varphi_n\mathrm{j}, \lambda_n\mathrm{j})$. here $\mathrm{p_k}$ and $\mathrm{p}_n\mathrm{j}$ refers to two specific position such as predicted and output location. Additionally, φ_k and φ_nj are the the longitudes of the positions, whereas λ_k and λ_nj are the latitudes of the positions. then, the node will be Assign to the closest vessel based on the LSTM model location prediction.

While the Multi-LSTM model demonstrates strong performance, it has a significant limitation.

The number of individual LSTM model is directly proportional to the number of vessels under study, leading to an elongated training period and increased computational cost. The limitation could be overcome by the following architectures.

3.3.2 1D CNN-LSTM Architecture

To address the track association problem, it is essential to devise a model that can effectively incorporate both temporal and spatial information. The CNN-LSTM model, as illustrated in Figure 3.6, represents a comprehensive solution that integrates one CNN layer and two LSTM layers to capture the underlying temporal and spatial dependencies present in AIS data. The architecture depicted in the figure 3.6 showcases the CNN-LSTM model, designed to extract salient features by incorporating an input layer following a 1D Convolutional layer.

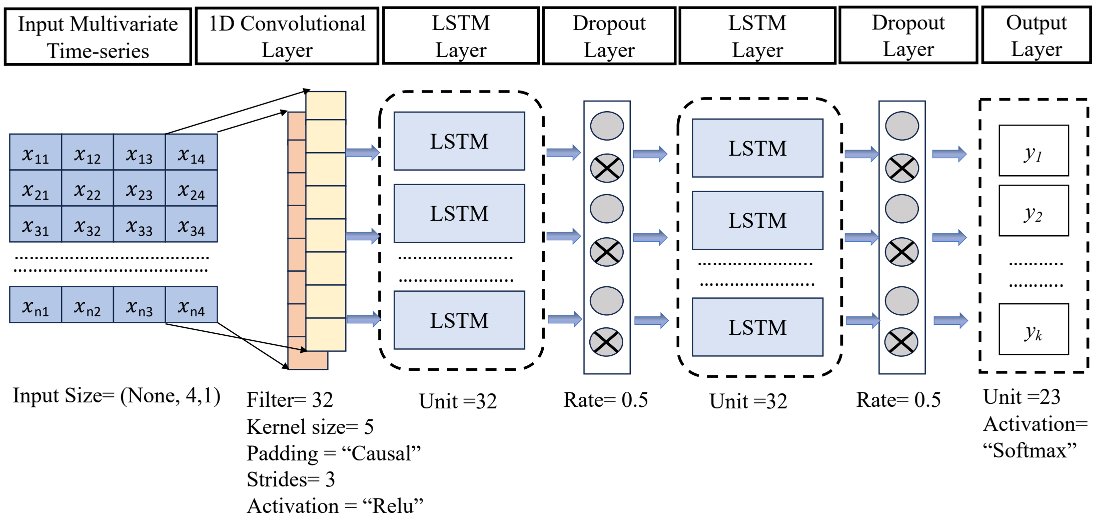


Figure 3.6: The proposed 1D CNN-LSTM architecture

Subsequently, the output from the convolutional layer undergoes further processing through LSTM layers, with the inclusion of two dropout layers after each LSTM layer. The parameter configurations for the CNN, LSTM, and Dropout layers are displayed at the bottom of each respective layer in the figure. Finally, the model concludes with an output layer in the form of a dense layer, containing a number of neurons equivalent to the count of vessels present in the training data.

The detailed step by step implementation for track association using one Dimensional CNN-LSTM is discussed below:

Step 1: The proposed hybrid architecture takes an input vector represented by

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & x_{n3} & x_{n4} \end{bmatrix}$$

with dimensions (None, 4, 1). The first dimension, denoted as "None," signifies the dynamic batch size commonly used in Keras implementation. Keras is a Python-based open-source neural network library that simplifies the prototyping of neural networks. The input vector consists of four variables, expressed as one-dimensional arrays.

Step 2: To extract salient features from the input, a 1D convolutional layer is constructed. The 1D CNN operates with a kernel that can traverse only in a single direction. The convolutional layer processes the input and generates an output of size (None, 2, 32).

Step 3 : Subsequently, the output from the convolutional layer undergoes further processing through LSTM layers, with the inclusion of two dropout layers after each LSTM layer. The parameter configurations for the CNN, LSTM, and dropout layers are displayed at the bottom of each respective layer in the figure.

Step 4 : Finally, the model concludes with an output layer in the form of a dense layer, containing a number of neurons equivalent to the count of vessels present in the training data.

3.3.3 Temporal-Graph Convolution Neural Network (T-GCN)

Temporal-Graph Neural convolutional Neural Network (T-GCNN) serves as an advanced extension of the conventional Graph Neural Network (GNN). Specifically designed to handle graph-structured data, the T-GCNN incorporates temporal dynamics to offer a more nuanced analysis. The Spatial Graph Convolutional Neural network help to capture spatial aspects of nodes in Graph created by the AIS dataset. However, The Graph structure created by the nodes and edges have temporal dependency as well. Acquiring the temporal dependence is another key problem in track association. The key idea of Spatio-temporal graph convolutional neural networks is to consider spatial dependency and temporal dependency at the same time. Many current approaches integrate graph convolutions to capture spatial dependency with RNNs or 1D CNNs to model the temporal dependency.

At present, the most widely used neural network model for processing sequence data is the

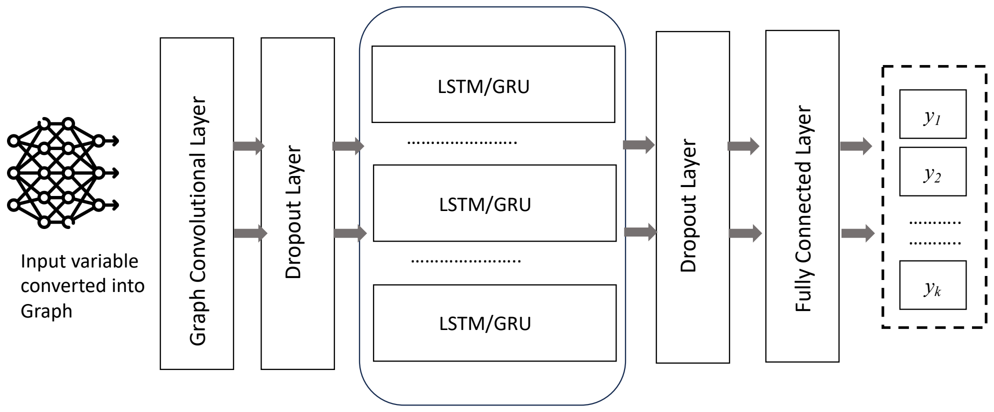


Figure 3.7: Proposed Temporal-Graph Convolutional Neural Network(T-GCNN

recurrent neural network (RNN). However, due to defects such as gradient disappearance and gradient explosion, the traditional recurrent neural network has limitations for long-term prediction [41] The LSTM model [61] and the GRU model [68] are variants of the recurrent neural network and have been proven to solve the above problems. The basic principles of the LSTM and GRU are roughly the same [69]. They all use gated mechanism to memorize as much long-term information as possible and are equally effective for various tasks. As illustrated in Figure 3.7, the architecture of the Temporal-GCN is specifically designed to manage the unique challenges posed by the AIS dataset. Initially, the input is transformed into a graph structure, which serves as the input to the Graph Convolutional Network (GCN) layer. A dropout layer follows the GCN, serving as a regularization mechanism to prevent overfitting. Subsequently, the output from the dropout layer is processed by a Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) layer. This stage is crucial for capturing the temporal dependencies inherent to the AIS data. Finally, a fully connected layer is appended to the network, its dimensions corresponding to the number of vessels present in the training set.

The detailed step by step implementation for track association using Temporal-Graph Convolution Neural Network is discussed below:

Step 1: We employ an unweighted graph $G = (V, E)$ to delineate the topological configuration of the AIS dataset, treating each timestamp as a node. In this context, V is a set of vessel nodes, $V = \{v_1, v_2, \ldots, v_N\}$, where N denotes the total number of nodes in the surveillance area. E constitutes the set of edges representing interactions or spatial proximities between vessels.

Step 2: The adjacency matrix A serves to encapsulate the connectivity between each nodes. Specifically, $A \in \mathbb{R}^{N\times N}$. This adjacency matrix contains only binary elements—0 and 1—to signify the existence or absence of a relationship between any two nodes. An element is set to 1 if the node belongs to a particular vessels, if it doesn't belong to a specific vessels then it is 0.

Step 3: To further enrich the representation of the AIS dataset, a feature matrix $X^{N\times P}$ is introduced. This matrix encapsulates vessel-related attributes, serving as node attributes in the network. Formally, $X \in \mathbb{R}^{N\times P}$, where P signifies the number of node attribute features, which include the latitude, longitude, speed and course related to each vessel.

Step 4: The Temporal-Graph Convolutional Neural Network (T-GCN) is initialized with predefined hyperparameters. The input layer is designed to accept sequences with a shape of $t \times P$, where t represents the number of timesteps and P denotes the feature dimension.

Step 5: Spatial features are convolved through a Graph Convolutional Network (GCN) layer. The adjacency matrix A of the graph G is utilized to capture the spatial correlation between nodes.

Step 6: A temporal layer is applied to the output of the GCN layer, capturing the temporal dependencies. This is typically achieved through 1D convolutional layers or LSTM layers or GRU layers.

Step 7: Finally, the model concludes with an output layer in the form of a dense layer, containing a number of neurons equivalent to the count of vessels present in the training data.

3.4 Performance Evaluation Metrics

To evaluate the effectiveness of our model, we generate the confusion matrix, which compares the actual classes with the predicted ones. The diagonal entries of the matrix represent the true positive predictions made by the model. However, relying solely on the confusion matrix is not enough to accurately quantify the performance of the model. Therefore, we compute several commonly used evaluation metrics to fully assess the model's performance, including:

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP}+\text{FN}} \tag{3.4.1}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN}+\text{FP}} \tag{3.4.2}$$

$$\text{F1 score} = \frac{\text{Sensitivity} + \text{Specificity}}{2} \tag{3.4.3}$$

$$\text{Accuracy} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{FP}+\text{FN}+\text{TN}} \tag{3.4.4}$$

The abbreviations TP, TN, FP, and FN correspond to true positive, true negative, false positive, and false negative predictions, respectively. Since the dataset has a class imbalance, we calculate the micro-average value for all metrics, which aggregates the contributions of all classes to determine the average metric.

CHAPTER 4

Results and Discussion

The results and discussion chapter is divided into four sections. In the initial section of the result analysis, we delve into the efficacy of the Multi-model Long Short-Term Memory (LSTM) framework, particularly focusing on its application to datasets of a smaller scale. This examination is accompanied by a detailed overview of the chosen hyperparameters. Subsequently, we present a comparative evaluation of performance metrics for both small and large-scale datasets, highlighting the proficiency of the 1D convolutional LSTM architecture. The rationale behind the selection of specific hyperparameters is also illustrated in this context. The third section evaluates the performance of the Temporal Graph Convolutional Neural Network, assessing its applicability and effectiveness across datasets varying in scale. The concluding section presents a comprehensive comparison between our hybrid neural network model and conventional neural network methodologies, thereby highlighting the relative improvement in performance achieved by our approach.

4.1 Multi-Model LSTM

To initiate the performance evaluation of the multi-model Long Short-Term Memory (LSTM), the evaluation begins with a dataset of relatively smaller magnitude, encompassing five distinct maritime vessels. The dataset utilized in this study consists of a total of 5,688 data points. These data points have been strategically chosen and employed for both training and validation of the model. An additional database containing 1,422 data points has been specifically set aside for the purpose of performing tests. The application of the model to the test dataset yielded results that can be described as notably promising. To provide a comprehensive analysis of the model's effectiveness in accurately categorizing different types of vessels, we present a thorough examination of the confusion matrix and evaluation metrics. The confusion matrix, depicted in

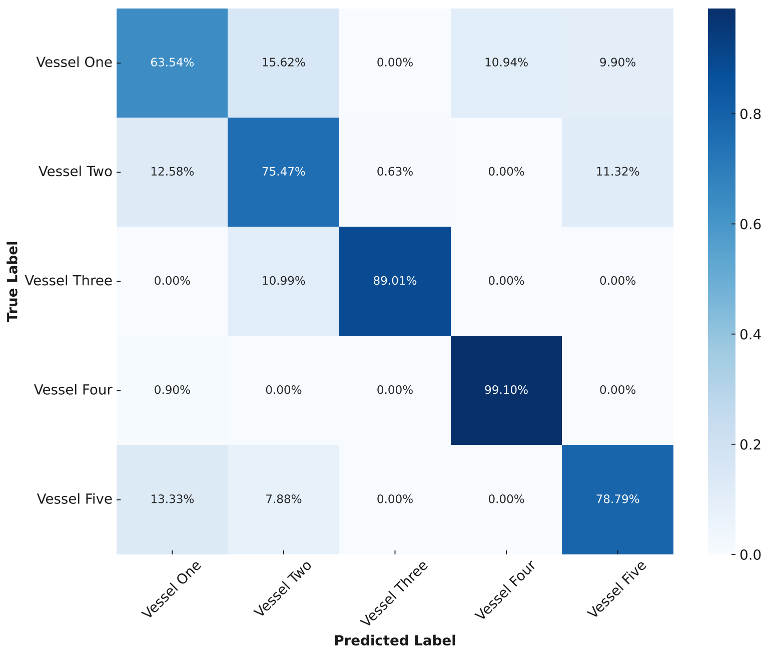


Figure 4.1: Confusion matrix for the multi model LSTM

Figure 4.1, offers a visual representation of the model's classification performance. It illustrates the number of instances where the model correctly or incorrectly predicted the vessel types. The model demonstrated commendable proficiency in accurate class prediction of the Vessels, achieving an average accuracy rate of 81% across the board.

	Accuracy	Precision	Recall	F1 score
Vessel One	0.696	0.725	0.670	0.70
Vessel Two	0.745	0.712	0.782	0.75
Vessel Three	0.942	1.000	0.890	0.94
Vessel Four	0.909	0.840	0.990	0.91
Vessel Five	0.792	0.826	0.760	0.79

Table 4.1: Evaluation metrics for the multi-model LSTM

Furthermore, Table 4.1 presents a detailed summary of the evaluation metrics employed to assess the model's performance. These metrics include precision, recall, and F1-score, which provide insights into the model's ability to correctly identify each vessel type. It is observed that the performance of the multi-LSTM model exhibits a degree of variability across different classes. Particularly, in scenarios where vessel tracks intersect or overlap, the model's predictive accuracy diminishes noticeably. For instance, the overlapping trajectories of Vessel 1 and Vessel 2 resulted in a reduced accuracy performance of 73% and 75%, respectively. In contrast, Vessels 4 and 5 displayed superior predictive accuracy, each achieving an accuracy rate of 94% and 91%. The hyperparameters are optimized using the random search method. The optimized hyperparameters are listed in table 4.2.

Hyperparameters	Value
Number of LSTM hidden cells	32
Return sequences for first layer	True
Number of skip connections	2
LSTM activation function	Relu
Batch size	10
Loss function	Mean Squared error
Learning rate	0.0001
Epochs	100

Table 4.2: Hyperparameters setting of the multi model LSTM

Multi-LSTM model for vessel trajectory classification requires constructing multiple classification models as the number of vessels increases. This introduces complexity in model development and inference. Additionally, incorporating new vessel trajectories necessitates training additional models on the new data, as the current approach relies on computing Haversine distance matrices between trajectories outside of the neural network. This matrix calculation imposes additional computational overhead.

To mitigate these challenges, we proposed several hybrid neural network architectures for joint modeling of all vessel trajectories. Convolutional layers in a hybrid neural network can extract spatial features from trajectory data. This eliminates the requirement for computation of distance matrices external to the model. A single model can be trained on trajectory data from all vessels and later fine-tuned to incorporate new trajectories. This reduces model complexity and computational burden relative to multi-model LSTM approaches. The proposed model aims to improve scalability while retaining the representational capacity for accurate track association.

4.2 1D Convolutional LSTM

To evaluate the performance of a 1D-CNN LSTM, at first, we start with a small-scale dataset consisting of five vessels same as the multi-model LSTM model.

The efficacy of the model is assessed through a confusion matrix, as depicted in Figure 4.2. The outcomes of this assessment for the small-scale model are notably accurate, except for one vessel that demonstrated an accuracy of 96.4%. This high level of precision can be ascribed to the advantageous combination of well-balanced class data and the limited quantity of vessels requiring classification.

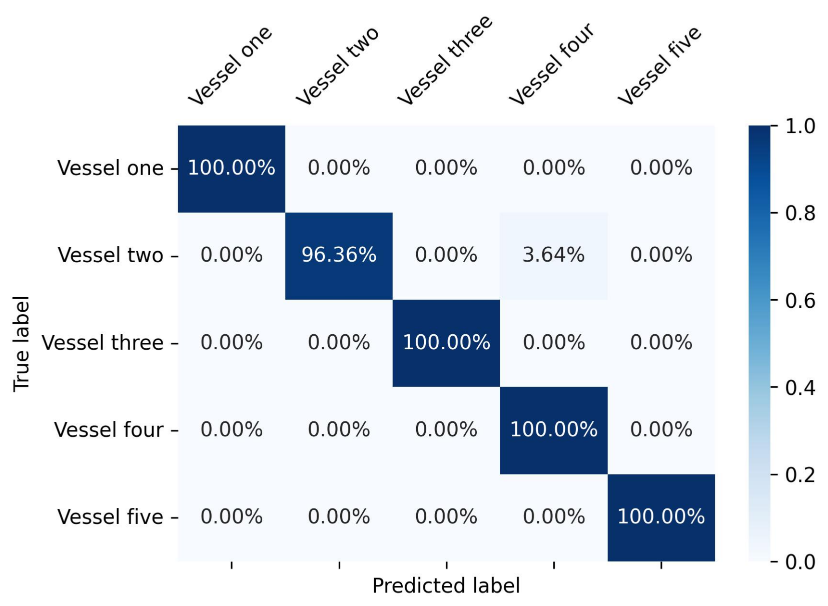


Figure 4.2: Confusion matrix implementing CNN-LSTM architecture for small scale dataset.

Further, to probe the model's capability to adapt and perform in varied settings, an extensive large-scale experiment was undertaken. The proposed hybrid Neural Network aimed to address two main challenges often seen in real-world situations: the difference in the amount and timing of data collected from different vessels, and the occurrence of paths crossing over each other. For this extensive analysis, we looked at 327 vessels, divided into 11 groups, with each group starting with around 30 vessels. After preprocessing the data, the final number of vessels was reduced to 264. The success of the model in this larger setting is measured using the 95% confidence interval from the results of all 11 groups, as calculated using Equation 4.2.1.

$$\text{Confidence Interval} = \bar{x} \pm \left(1.96 \times \frac{s}{\sqrt{n}}\right) \tag{4.2.1}$$

where $\bar{x}$ is the sample mean, s is the standard deviation of the sample, n is the sample size, and 1.96 is the z-score from the standard normal distribution corresponding to a 95% confidence level. The standard error of the mean (SEM) is calculated as $\frac{s}{\sqrt{n}}$. Table 4.3 presents the evaluation metrics for a deep learning architecture, specifically the CNN-LSTM model, across four principal indicators: Accuracy, Precision, Recall, and F1 Score. Accompanying each metric is the Standard Error of the Mean (SEM), which serves to quantify the estimate's uncertainty. An accuracy of 0.89 ± 0.02 is reported for the CNN-LSTM model, indicating a 95% confidence interval about the mean accuracy score. Precision and f1 score are similarly reported as 0.89 ± 0.02, while recall is 0.91 ± 0.02.

Table 4.3: Evaluation matrix for the deep learning architectures.

Deep Learning Models	Accuracy	Precision	Recall	F1 Score
CNN-LSTM	0.89 ± 0.02	0.89 ± 0.02	0.91 ± 0.02	0.89 ± 0.02

This research offers a crucial understanding of how the 1D-CNN LSTM model can be adapted for use in complicated, real-life situations, underlining its usefulness for practical implementations. The intricacies involved in the thorough assessment of the 1D-CNN LSTM model are depicted in figure 4.3, where vessels from the first batch that lack specific identifiable details are shown as white dots. This includes cases where vessel paths overlap without clear markers for their beginning and ending points.

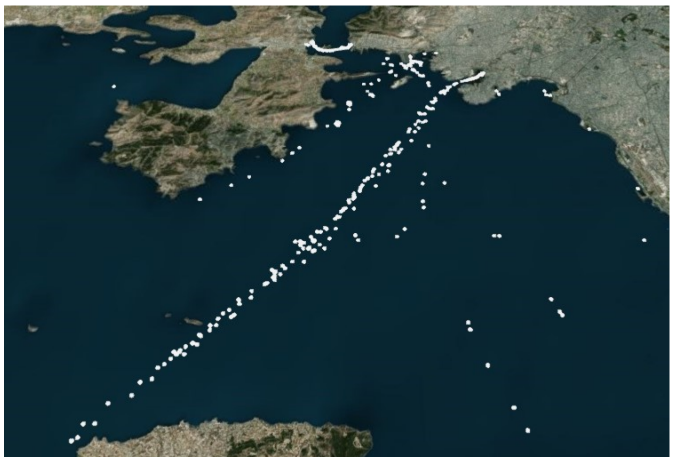


Figure 4.3: Unclassified vessels

However, as demonstrated in figure 4.4, our model is capable of accurately detecting these vessels and recovering their tracks, even in cases where navigation routes overlap. The exceptional performance of 1D CNN-LSTM is also evident from the stability of its AUC curve for the training and validation loss and accuracy, as shown in figure 4.5. The model's performance can also be quantitatively assessed by comparing its predictions with the true labels.

To identify the optimal blend of CNN and LSTM layers, various configurations were tested. We have found that 1 CNN layer and two LSTM layers outperform the other combination with approximately 90% accuracy (see in figure 4.6). The improvement of the model depends on hyperparameter tuning. Parameters such as the number of filters in the convolutional layer, the size and stride of the convolutional kernel, and the number of hidden cells in the LSTM layer are optimized using the random search method [70] (as specified in Table 4.4). The optimization of these hyperparameters spans numerous levels, leading to a computationally demanding and time-intensive task. This is due to the vast number of experiments generated by the combination

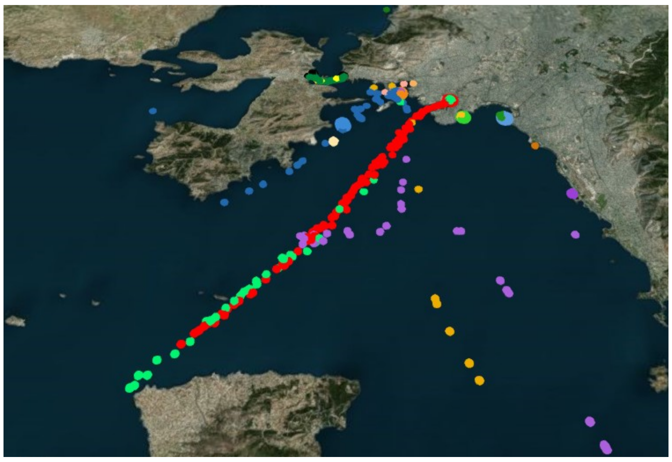


Figure 4.4: Track association using the proposed algorithm

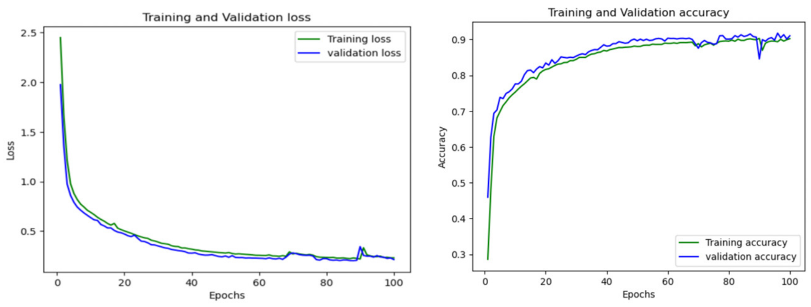


Figure 4.5: Training and validation loss and accuracy curve for CNN-LSTM architecture.

of these hyperparameters. The results for the different hyperparameters are shown in the figure 4.7.

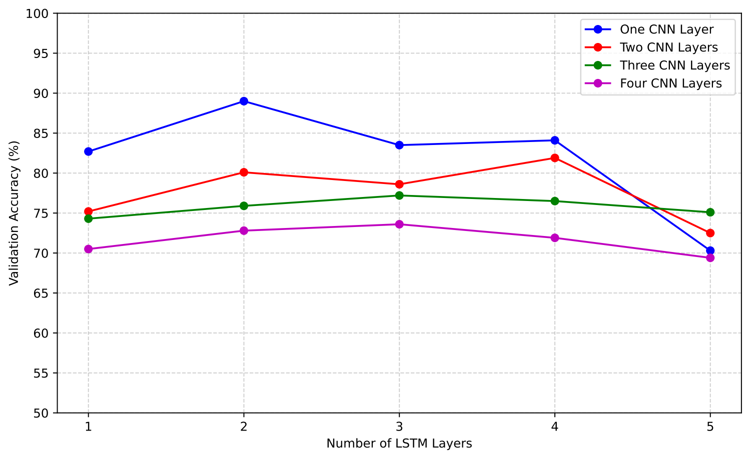


Figure 4.6: Ablation study for finding the optimized combination of CNN layers and LSTM layers. The combination of one CNN layer and two LSTM layers outperforms the other in terms of validation accuracy score.

Table 4.4: Hyperparameter tuning for the hybrid 1D CNN LSTM architecture.

Parameters	**Value**
Convolutional layer filters	32
Convolutional kernel size	5
Convolutional kernel stride	3
Convolutional layer activation function	ReLU
Convolutional layer padding	Causal
Number of LSTM hidden cells	32
Number of skip connections	2
LSTM activation function	Sigmoid
Batch size	100
Loss function	Categorical cross entropy
Learning rate	0.0001
Epochs	100

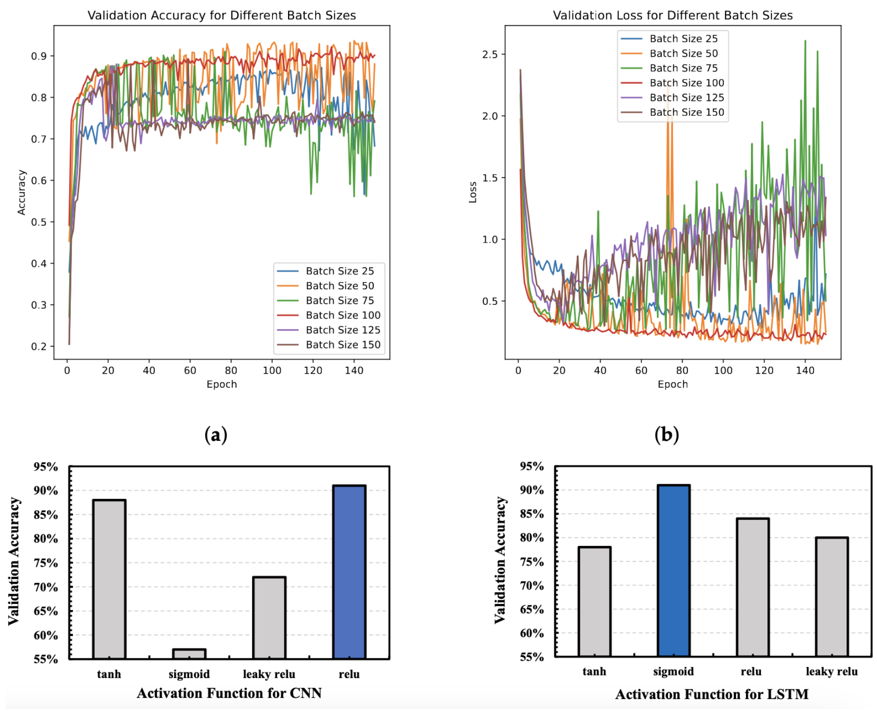


Figure 4.7: Selecting the optimal hyperparameters using the grid search method. (a) Validation accuracy for different batch sizes; (b) Validation loss for different batch sizes; (c) Performance of different activation functions in CNN layer; (d) Performance of different activation functions in LSTM layers.

4.3 Temporal GCN

Temporal graph convolutional networks (TGCNs) have demonstrated exceptional ability for capturing sequential and spatial behaviors in time series data [71]. To implement TGCNs for multi-object track association, we represent each object id as a node v_i in a graph $G = (V, E)$. Edges $e_{ij} \in E$ connect nodes v_i and v_j if the corresponding tracks belong to the same vessel. The goal is to predict these edges, which depend on whether a pair of nodes/tracks are associated with the same vessel. We encode each node v_i with features x_i consisting of timestamp, longitude, latitude, speed, and course. Edges e_{ij} are encoded with a temporal feature τ_{ij} representing the time difference between nodes v_i and v_j.

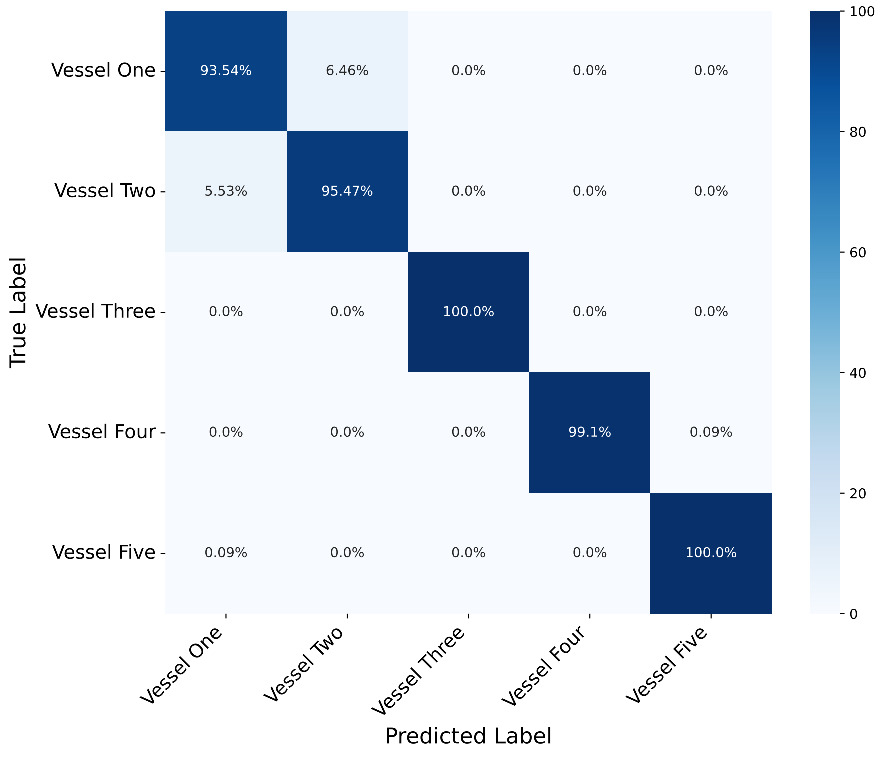


Figure 4.8: Confusion matrix implementing Temporal GCN architecture for a small-scale dataset.

We trained the Temporal GCN on the same small dataset. The performance evaluation of our model, as illustrated in figure 4.8, demonstrates outstanding predictive accuracy across five distinct vessel categories. The model exhibits exceptional precision with Vessel Three and Vessel Five, achieving a perfect classification rate of 100%. Similarly, Vessel Four shows near-perfect accuracy with 99.1% of predictions being correct. Vessel One and Vessel Two also display robust classification capabilities with 93.54% and 95.47% accuracy, respectively. Further tuning of the model architecture and training process could improve multi-object discrimination for intersecting tracks.

While the temporal graph convolutional network (TGCN) demonstrated impressive results on a smaller dataset, its effectiveness diminished when applied to a larger dataset consisting of 264 ship trajectories divided into 11 groups. The TGCN-LSTM model managed to attain an average accuracy of 78%, but this was accompanied by a 4% variation across the different groups (see in table 4.5). This pattern suggests that the T-GCN's performance is influenced by the total number of data points in each group, showing a decrease in efficiency as the number of tracks in each group increases.

Table 4.5: Evaluation metrics for the Temporal GCN architectures.

Deep Learning Models	Accuracy	Precision	Recall	F1 Score
Temporal GCN (LSTM)	0.78 ± 0.04	0.77 ± 0.03	0.78 ± 0.04	0.78 ± 0.04
Temporal GCN (GRU)	0.73 ± 0.03	0.75 ± 0.03	0.75 ± 0.03	0.74 ± 0.03

The large number of training parameters of the Temporal Graph Convolutional Network (T-GCN), which increases in size with the dataset, likely led to issues of instability and under-performance. This was also evident in the case of a T-GCN combined with a Gated Recurrent Unit (GRU) model, referred to as T-GCN(GRU). The overall effectiveness of the T-GCN(GRU) model was subpar, achieving an accuracy of 73% and an F1 score of 74%. The hyperparameters used for the optimized performance are listed here in the table 4.2.

Table 4.6: Hyperparameter tuning for the T-GCN (LSTM) architecture.

Parameters	Value
GCN output size	32
Number of LSTM units	64
Optimization algorithm	Adam
Loss function	Sparse categorical cross entropy
Model performance metrics	Accuracy
Number of training epochs	100
Fraction of data for validation	0.15
Batch Size	100
Length of input sequences	10

Scaling up temporal graph networks likely requires more robust regularization and optimization methods. Further analysis is needed to adapt temporal graph networks to large real-world transportation datasets.

4.4 Overall Comparison

Through a comprehensive quantitative approach, we conducted an evaluation of the performance of nine alternative deep learning architectures. The following architectures are being studied: 1D CNN LSTM, Temporal GCN (LSTM), Temporal GCN (GRU), Conventional Convolutional Neural Network (CNN), long short-term memory (LSTM), Bi-directional LSTM (Bi-LSTM) [48, 49], Bi-directional gated recurrent unit (Bi-GRU) [50], and the feedforward artificial neural network (ANN). All of the models have been tested using the same large-scale dataset, which consists of 264 vessels that are split into eleven batches. The models are tested to see their versatility to handle the large dataset, their overall performance, and the variability of performance among the batches. The comparative analysis presented in table 4.7 clearly delineates the performance hierarchy among the evaluated deep learning architectures.

Table 4.7: Comparison of performance metrics for the deep learning architectures

Deep Learning Models	Accuracy ± SEM	Precision ± SEM	Recall ± SEM	F1 Score ± SEM
CNN-LSTM	0.89 ± 0.02	0.89 ± 0.02	0.91 ± 0.02	0.89 ± 0.02
Bi-LSTM	0.87 ± 0.02	0.86 ± 0.02	0.86 ± 0.02	0.86 ± 0.02
Bi-GRU	0.83 ± 0.03	0.83 ± 0.03	0.84 ± 0.03	0.83 ± 0.03
LSTM	0.80 ± 0.03	0.80 ± 0.03	0.81 ± 0.03	0.80 ± 0.03
Temporal GCN (LSTM)	0.78 ± 0.04	0.77 ± 0.03	0.78 ± 0.04	0.78 ± 0.04
CNN	0.74 ± 0.04	0.74 ± 0.04	0.74 ± 0.04	0.73 ± 0.04
Temporal GCN (GRU)	0.73 ± 0.03	0.75 ± 0.03	0.75 ± 0.03	0.74 ± 0.03
ANN	0.46 ± 0.05	0.46 ± 0.05	0.45 ± 0.05	0.48 ± 0.05

The CNN-LSTM model emerges as the front-runner, attaining an exemplary accuracy of 0.89 ± 0.02, matched by its precision and underscored by a superior recall of 0.91 ± 0.02, culminating in an F1 score of 0.89 ± 0.02. This performance is not only consistent across metrics but also signifies the robustness of the model, as reflected by the low standard error of the mean (SEM).

Trailing closely, the Bi-LSTM and Bi-GRU models exhibited commendable performances with accuracies of 0.87 ± 0.02 and 0.83 ± 0.03, respectively. These bidirectional models, by virtue of their architecture, effectively captured the sequential dependencies in both forward and reverse directions, which is instrumental in complex temporal pattern recognition tasks.

The LSTM model, with its accuracy of 0.80 ± 0.03, stands out for its ability to surpass more conventional models like the CNN, which recorded an accuracy of 0.74 ± 0.04. This highlights the LSTM's capacity to capture long-term dependencies, a critical aspect often missing in traditional CNNs.

The Temporal Graph Convolutional Network (GCN) models, when integrated with LSTM and

GRU units for sequence processing, exhibited promising outcomes despite not achieving peak performance. These models, with accuracies of 0.78 ± 0.04 for the TGCN-LSTM and 0.73 ± 0.03 for the TGCN-GRU, demonstrate the effective combination of graph-based neural networks with temporal processing units. This integration marks a significant advancement in the field, opening up new possibilities for handling structured time series data.

In stark contrast, the ANN model lagged significantly behind, with the lowest accuracy of 0.46 ± 0.05, a clear indication of its inability to model the spatial and temporal dynamics inherent to the dataset. This reinforces the critical importance of choosing architectures that align with the complex nature of the data and the task at hand.

The superior performance of the 1D CNN-LSTM architecture can be ascribed to its distinctive ability to extract salient features while concurrently considering the sequential nature of the AIS dataset, which is essential for accurate prediction in time series datasets. The architecture takes into account both the spatial and temporal characteristics of the dataset, resulting in exceptional performance in all performance evaluation metrics. Taken together, these findings underscore the efficacy of deep learning models in addressing intricate maritime problems and emphasizing the potential of the 1D CNN-LSTM architecture for future tracking applications.

CHAPTER 5

Conclusion

This research demonstrates the efficacy of neural networks in solving the track association problem, which traditionally relies on physics-based methods. These methods often face constraints due to their dependence on only the preceding node to estimate the likelihood of the succeeding node's path. To overcome such limitations, this study employs neural network-based sequential models designed to encapsulate the spatio-temporal dynamics within the data. A noteworthy strategy is the multi-model LSTM technique, which, despite its effectiveness, encounters a decrease in computational efficiency as the count of vessels—and consequently, the number of required models—grows. Graph Neural Networks (GNNs) offer an alternative by representing nodes within a graph structure, creating distinct graphs for individual vessels. However, the growth in model parameters with increasing complexity has been observed to cause overfitting, which compromises the model's predictive power. Among the tested architectures, the 1D CNN-LSTM model distinguishes itself by its adeptness at assimilating spatial and temporal data, particularly within the AIS dataset context.

The study acknowledges the unexplored avenue of integrating novel vessels into the tracking system. Future developments could potentially introduce an innovative mechanism to track nodes related to new vessels, integrating them effectively into the existing framework. While the temporal GCN has shown promise in handling datasets of limited size, optimizing and refining the model's architecture could substantially augment its applicability and performance.

Bibliography

[1] Imtiaz Ahmed, Mikyoung Jun, and Yu Ding. *A Spatio-temporal Track Association Algorithm Based on Marine Vessel Automatic Identification System Data*. arXiv:2010.15921 [cs]. June 2022. URL: http://arxiv.org/abs/2010.15921 (visited on 09/06/2023).

[2] *What is the Automatic Identification System (AIS)?* en-US. Aug. 2021. URL: https://help.marinetraffic.com/hc/en-us/articles/204581828-What-is-the-Automatic-Identification-System-AIS- (visited on 09/06/2023).

[3] Jessica H Ford et al. "Detecting suspicious activities at sea based on anomalies in Automatic Identification Systems transmissions". In: *PLoS One* 13.8 (2018), e0201640.

[4] Ziqiang Ou and Jianjun Zhu. "AIS database powered by GIS technology for maritime safety and security". In: *The Journal of Navigation* 61.4 (2008), pp. 655–665.

[5] Duong Nguyen and Ronan Fablet. *TrAISformer-A generative transformer for AIS trajectory prediction*. arXiv:2109.03958 [cs]. May 2023. URL: http://arxiv.org/abs/2109.03958 (visited on 09/06/2023).

[6] Md Rabiul Hasan, Zhichao Liu, and Asif Rahman. "Energy Consumption Modeling for DED-based Hybrid Additive Manufacturing". In: *arXiv preprint arXiv:2305.09104* (2023).

[7] L. Zhang et al. "Continuous tracking of targets for stereoscopic hfswr based on imm filtering combined with elm". In: *Remote Sensing* 12 (2 2020), p. 272. DOI: 10.3390/rs12020272.

[8] Ujwalla Gawande, Kamal Hajari, and Yogesh Golhar. "Pedestrian detection and tracking in video surveillance system: issues, comprehensive review, and challenges". In: *Recent Trends in Computational Intelligence* (2020), pp. 1–24.

[9] S. Nakayama et al. "Comparing spatial patterns of marine vessels between vessel-tracking data and satellite imagery". In: *Frontiers in Marine Science* 9 (2023). DOI: 10.3389/fmars.2022.1076775.

[10] Liangbin Zhao, Guoyou Shi, and Jiaxuan Yang. "Ship trajectories pre-processing based on AIS data". In: *The Journal of Navigation* 71.5 (2018), pp. 1210–1230.

[11] Yanjie Zeng et al. "Robust multivehicle tracking with wasserstein association metric in surveillance videos". In: *IEEE Access* 8 (2020), pp. 47863–47876.

[12] Mousa Nazari, Saeid Pashazadeh, and Leyli Mohammad-Khanli. "An adaptive density-based fuzzy clustering track association for distributed tracking system". In: *IEEE Access* 7 (2019), pp. 135972–135981.

[13] Jayaramu Raghu et al. "Comprehensive track segment association for improved track continuity". In: *IEEE Transactions on Aerospace and Electronic Systems* 54.5 (2018), pp. 2463–2480.

[14] Lan You et al. “St-seq2seq: A spatio-temporal feature-optimized seq2seq model for short-term vessel trajectory prediction”. In: *IEEE Access* 8 (2020), pp. 218565–218574.

[15] Md Rabiul Hasan and Thorsten Wuest. “A Review of Sustainable Composites Supply Chains”. In: *IFIP International Conference on Advances in Production Management Systems*. Springer. 2022, pp. 448–455.

[16] Diego M Jiménez-Bravo et al. “Multi-Object Tracking in Traffic Environments: A Systematic Literature Review”. In: *Neurocomputing* (2022).

[17] Samuel S Blackman. “Multiple hypothesis tracking for multiple target tracking”. In: *IEEE Aerospace and Electronic Systems Magazine* 19.1 (2004), pp. 5–18.

[18] Yaakov Bar-Shalom and Xiao-Rong Li. *Multitarget-multisensor tracking: principles and techniques*. Vol. 19. YBs Storrs, CT, 1995.

[19] Donald Reid. “An algorithm for tracking multiple targets”. In: *IEEE transactions on Automatic Control* 24.6 (1979), pp. 843–854.

[20] Su Pang and Hayder Radha. “Multi-Object Tracking Using Poisson Multi-Bernoulli Mixture Filtering For Autonomous Vehicles”. In: *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2021, pp. 7963–7967. DOI: 10.1109/ICASSP39728.2021.9415072.

[21] Samuele Capobianco et al. “Deep learning methods for vessel trajectory prediction based on recurrent neural networks”. In: *IEEE Transactions on Aerospace and Electronic Systems* 57.6 (2021), pp. 4329–4346.

[22] Robert A Best and JP Norton. “A new model and efficient tracker for a target with curvilinear motion”. In: *IEEE Transactions on Aerospace and Electronic Systems* 33.3 (1997), pp. 1030–1037.

[23] Derek Caveney. “Numerical integration for future vehicle path prediction”. In: *2007 American Control Conference*. IEEE. 2007, pp. 3906–3912.

[24] Emil Semerdjiev and Ludmila Mihaylova. “Variable-and fixed-structure augmented interacting multiple model algorithms for manoeuvring ship tracking based on new ship models”. In: *International Journal of Applied Mathematics and Computer Science* 10.3 (2000), pp. 591–604.

[25] Ameer Khan, Cees Bil, and Kaye E Marion. “Ship motion prediction for launch and recovery of air vehicles”. In: *Proceedings of OCEANS 2005 MTS/IEEE*. IEEE. 2005, pp. 2795–2801.

[26] S Bartelmaos, K Abed-Meraim, and S Attallah. “Fast algorithms for minor component analysis”. In: *IEEE/SP 13th Workshop on Statistical Signal Processing, 2005*. IEEE. 2005, pp. 239–244.

[27] Dezhong Peng and Zhang Yi. “A new algorithm for sequential minor component analysis”. In: *International Journal of Computational Intelligence Research* 2.2 (2006), pp. 207–215.

[28] Ugur Simsir and Seniz Ertugrul. “Prediction of manually controlled vessels’ position and course navigating in narrow waterways using Artificial Neural Networks”. In: *Applied Soft Computing* 9.4 (2009), pp. 1217–1224.

[29] Joshua Joseph et al. “A Bayesian nonparametric approach to modeling motion patterns”. In: *Autonomous Robots* 31.4 (2011), p. 383.

[30] Giuliana Pallotta et al. “Context-enhanced vessel prediction based on Ornstein-Uhlenbeck processes using historical AIS traffic patterns: Real-world experimental results”. In: *17th International Conference on Information Fusion (FUSION)*. IEEE. 2014, pp. 1–7.

[31] Xiu-Rong Guo et al. “An improved neural network based fuzzy self-adaptive Kalman filter and its application in cone picking robot”. In: *2009 International Conference on Machine Learning and Cybernetics*. Vol. 1. IEEE. 2009, pp. 573–577.

[32] Lokukaluge P Perera and Carlos Guedes Soares. “Ocean vessel trajectory estimation and prediction based on extended Kalman filter”. In: *The Second International Conference on Adaptive and Self-Adaptive Systems and Applications*. 2010, pp. 14–20.

[33] Andrzej Stateczny and Witold Kazimierski. “Multisensor Tracking of Marine Targets: Decentralized Fusion of Kalman and Neural Filters”. In: *International Journal of Electronics and Telecommunications* 57 (2011), pp. 65–70.

[34] Lokukaluge P Perera, Paulo Oliveira, and C Guedes Soares. “Maritime traffic monitoring based on vessel detection, tracking, state estimation, and trajectory prediction”. In: *IEEE Transactions on Intelligent Transportation Systems* 13.3 (2012), pp. 1188–1200.

[35] Biϕrnar R Dalsnes et al. “The neighbor course distribution method with Gaussian mixture models for AIS-based vessel trajectory prediction”. In: *2018 21st International Conference on Information Fusion (FUSION)*. IEEE. 2018, pp. 580–587.

[36] Enmei Tu et al. “Modeling Historical AIS Data For Vessel Path Prediction: A Comprehensive Treatment”. In: *arXiv preprint arXiv:2001.01592* (2020).

[37] Angelos Valsamis et al. “Employing traditional machine learning algorithms for big data streams analysis: The case of object trajectory prediction”. In: *Journal of Systems and Software* 127 (2017), pp. 249–257.

[38] Rui Chen et al. “Predicting future locations of moving objects by recurrent mixture density network”. In: *ISPRS International Journal of Geo-Information* 9.2 (2020), p. 116.

[39] Tamara A Volkova, Yulia E Balykina, and Alexander Bespalov. “Predicting ship trajectory based on neural networks using AIS data”. In: *Journal of Marine Science and Engineering* 9.3 (2021), p. 254.

[40] Huanhuan Li, Hang Jiao, and Zaili Yang. “AIS data-driven ship trajectory prediction modelling and analysis based on machine learning and deep learning methods”. In: *Transportation Research Part E: Logistics and Transportation Review* 175 (2023), p. 103152.

[41] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. “Learning long-term dependencies with gradient descent is difficult”. In: *IEEE transactions on neural networks* 5.2 (1994), pp. 157–166.

[42] Piotr Borkowski. “The ship movement trajectory prediction algorithm using navigational data fusion”. In: *Sensors* 17.6 (2017), p. 1432.

[43] Huang Tang, Yong Yin, and Helong Shen. “A model for vessel trajectory prediction based on long short-term memory neural network”. In: *Journal of Marine Engineering & Technology* 21.3 (2022), pp. 136–145.

[44] Huarong Zheng, Rudy R Negenborn, and Gabriel Lodewijks. “Trajectory tracking of autonomous vessels using model predictive control”. In: *IFAC Proceedings Volumes* 47.3 (2014), pp. 8812–8818.

[45] Wided Hammedi, Bouziane Brik, and Sidi Mohammed Senouci. “Toward optimal MEC-based collision avoidance system for cooperative inland vessels: a federated deep learning approach”. In: *IEEE Transactions on Intelligent Transportation Systems* (2022).

[46] Gözde Boztepe Karataş, Pinar Karagoz, and Orhan Ayran. “Trajectory pattern extraction and anomaly detection for maritime vessels”. In: *Internet of Things* 16 (2021), p. 100436.

[47] Yongfeng Suo et al. “A ship trajectory prediction framework based on a recurrent neural network”. In: *Sensors* 20.18 (2020), p. 5133.

[48] Cheng-Hong Yang et al. “AIS-based intelligent vessel trajectory prediction using bi-LSTM”. In: *IEEE Access* 10 (2022), pp. 24302–24315.

[49] Jinwan Park, Jungsik Jeong, and Youngsoo Park. “Ship trajectory prediction based on bi-LSTM using spectral-clustered AIS data”. In: *Journal of Marine Science and Engineering* 9.9 (2021), p. 1037.

[50] Chang Wang, Hongxiang Ren, and Haijiang Li. “Vessel trajectory prediction based on AIS data and bidirectional GRU”. In: *2020 International Conference on Computer Vision, Image and Deep Learning (CVIDL)*. 2020, pp. 260–264. DOI: 10.1109/CVIDL51233.2020.00-89.

[51] Saeed Mehri, Ali Asghar Alesheikh, and Anahid Basiri. “A contextual hybrid model for vessel movement prediction”. In: *IEEE Access* 9 (2021), pp. 45600–45613.

[52] Da-wei Gao et al. “A novel MP-LSTM method for ship trajectory prediction based on AIS data”. In: *Ocean Engineering* 228 (2021), p. 108956.

[53] Wells Wang et al. “A multi-task learning-based framework for global maritime trajectory and destination prediction with AIS data”. In: *Maritime Transport Research* 3 (2022), p. 100072.

[54] Brian Murray and Lokukaluge Prasad Perera. “An AIS-based deep learning framework for regional ship behavior prediction”. In: *Reliability Engineering & System Safety* 215 (2021), p. 107819.

[55] Cheng-Hong Yang et al. “Deep Learning for Vessel Trajectory Prediction Using Clustered AIS Data”. In: *Mathematics* 10.16 (2022), p. 2936.

[56] Yingkun Xu et al. “Deep learning for multiple object tracking: a survey”. In: *IET Computer Vision* 13.4 (2019), pp. 355–368.

[57] Cheng Zhong et al. “Inland ship trajectory restoration by recurrent neural network”. In: *The Journal of Navigation* 72.6 (2019), pp. 1359–1377.

[58] Xinyu Wang and Yingjie Xiao. “A Deep Learning Model for Ship Trajectory Prediction Using Automatic Identification System (AIS) Data”. In: *Information* 14.4 (2023), p. 212.

[59] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[60] Wikipedia. *Convolutional neural network*. accessed: December 22, 2022. 2022. URL: https://en.wikipedia.org/wiki/Convolutionalneuralnetwork.

[61] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[62] Guo Xie et al. “Motion trajectory prediction based on a CNN-LSTM sequential model”. In: *Science China Information Sciences* 63.11 (2020), pp. 1–21.

[63] Colah. *Understanding LSTM Networks*. accessed: December 22, 2022. 2015. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[64] Dogan Altan et al. “Discovering Gateway Ports in Maritime Using Temporal Graph Neural Network Port Classification”. In: *arXiv preprint arXiv:2204.11855* (2022).

[65] Zonghan Wu et al. “A comprehensive survey on graph neural networks”. In: *IEEE transactions on neural networks and learning systems* 32.1 (2020), pp. 4–24.

[66] Felix Wu et al. “Simplifying graph convolutional networks”. In: *International conference on machine learning*. PMLR. 2019, pp. 6861–6871.

[67] *Distance on a sphere: The Haversine Formula*. en. Oct. 2017. URL: https://community.esri.com/t5/coordinate-reference-systems-blog/distance-on-a-sphere-the-haversine-formula/ba-p/902128 (visited on 09/17/2023).

[68] Kyunghyun Cho et al. “On the properties of neural machine translation: Encoder-decoder approaches”. In: *arXiv preprint arXiv:1409.1259* (2014).

[69] Junyoung Chung et al. “Empirical evaluation of gated recurrent neural networks on sequence modeling”. In: *arXiv preprint arXiv:1412.3555* (2014).

[70] James Bergstra and Yoshua Bengio. “Random search for hyper-parameter optimization.” In: *Journal of machine learning research* 13.2 (2012).

[71] Ling Zhao et al. “T-gcn: A temporal graph convolutional network for traffic prediction”. In: *IEEE transactions on intelligent transportation systems* 21.9 (2019), pp. 3848–3858.