Ship Movement Prediction in Maritime Traffic using Machine Learning Methods

University of Turku
Department of Computing
Master of Science Thesis
Data Analytics
July 2026
Vera Salminen

Supervisors:
Farshad Farahnakian

UNIVERSITY OF TURKU
Department of Computing

VERA SALMINEN: Ship Movement Prediction in Maritime Traffic using Machine Learning Methods

Master of Science Thesis, 63 p.
Data Analytics
July 2026

Accurate ship trajectory prediction based on Automatic Identification System (AIS) data is essential for modern maritime safety, maritime traffic management, and autonomous vessel navigation. With the rise of Maritime Autonomous Surface Ships (MASS), improving maritime safety has become increasingly critical, with trajectory prediction playing a vital role in collision avoidance systems. Because AIS is widely adopted, most modern prediction approaches rely on forecasting future movements from historical trajectory data.

Various deep learning architectures have been explored for ship trajectory prediction. This thesis focuses on evaluating four specific deep learning models: Bidirectional Long Short-Term Memory (Bi-LSTM), Gated Recurrent Unit (GRU), Temporal Convolutional Network (TCN), and Temporal Fusion Transformer (TFT).

The primary objective of this thesis is to hyperparameter-optimize and compare these four models when predicting ship trajectories in the Baltic Sea. Regarding optimization, the analysis focuses on how unit and dropout values affect model performance. For performance evaluation, Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R^2 Score are used to evaluate variations within the same architecture as well as across different architectures.

The experimental results highlight specific scenarios where models failed to predict accurate trajectories. These failures were attributed to several factors, including data selection choices regarding both timeframe and vessel types, as well as the use of unsuitable training loss criteria such as MAE.

Keywords: Maritime situational awareness; ship trajectory prediction; AIS data; deep learning; hyperparameter optimization

Contents

List of Figures

Declaration on the Use of Artificial Intelligence

In the course of preparing this thesis, artificial intelligence tools, including ChatGPT, were employed to enhance grammar, academic style, clarity, and overall readability. These tools facilitated the revision of sentences, the structuring of explanations, and the clarification of technical concepts. Nevertheless, the design of the study, implementation of the pipeline, execution of experiments, interpretation of results, and formulation of final conclusions were conducted exclusively by the author. The author assumes full responsibility for the content of the thesis and has ensured that all outputs generated with the assistance of artificial intelligence were critically reviewed, modified, and verified prior to their inclusion.

1 Introduction

1.1 Maritime Traffic

The amount of Earth's surface covered by water is around 71%. [1] Around 90% of the global trade volume is handled by shipping, making it crucial to the world economy. [2] The volume of international trade has increased by a large amount due to the global economy's persistent growth. This has resulted in a rapid increase in vessel capacity, size, and sailing speed. [3]

Maritime traffic accidents can cause a large amount of property and environmental damage. [1] Since the introduction of Maritime Autonomous Surface Ships (MASS) and hybrid traffic, new safety issues and challenges are affecting the maritime transport sector. To prevent significant casualties and damages, it is important to explore new methods to ensure maritime safety. [2] Collision avoidance is one of the biggest challenges for the potential of autonomous vessels. Maritime Collision Avoidance Systems (MCAS) are used to give early warning signals to approaching vessels to avoid serious accidents. Ship trajectory prediction is a fundamental and crucial aspect of MCAS. Reliable forecasting of ship trajectories has many additional benefits for the shipping industry outside of collision avoidance. Examples of these include route planning, traffic management, port operations, and traffic anomaly detection. [3]

The development of digitalization, cloud computing, big data mining, Artificial

Intelligence (AI), and the Internet of Things (IoT) has started an era of intelligent shipping. Intelligent shipping requires high-quality data and data mining technologies, including data collection, preprocessing, compression, feature engineering, and network computing. To realize the potential of MASS navigation, real-time data mining and prediction are needed. [2]

1.2 Deep Learning in Ship Trajectory Prediction in the Baltic Sea

1.2.1 Research Aims

The primary objective of this thesis is to evaluate and compare four different deep learning architectures for ship trajectory prediction using AIS data in the Baltic Sea. The architectures selected for this study are Bidirectional Long Short-Term Memory (Bi-LSTM), Temporal Convolutional Network (TCN), Gated Recurrent Unit (GRU), and the Temporal Fusion Transformer (TFT).

While recurrent networks like GRU and Bi-LSTM are widely used for sequential data, this work evaluates them alongside TCN and TFT, which are less established in maritime literature. The benchmarking process focuses on how two key structural hyperparameters—the number of hidden units and the dropout rate—affect model performance. Each architecture is systematically evaluated across a predefined hyperparameter space, varying the hidden units ($4, 8, 16, 32, 64$, and 128) and dropout values (0.0 and 0.5) to determine the optimal configuration.

1.2.2 Research Questions

To achieve these aims, this thesis addresses the following research questions:

RQ1: What is the optimal combination of hidden units and dropout for each of the

four models?

RQ2: Which of the four optimized models demonstrates the best overall performance for ship trajectory prediction?

1.3 Thesis Structure

Chapter 2 presents a literature review focusing on the deep learning methods utilized in this thesis, alongside their prevalence in current research. It also details alternative methodologies for ship trajectory prediction, including traditional machine learning models and other deep learning architectures not implemented in this work. The primary focus of this chapter is to provide basic explanations of these various methods and to illustrate the methodological variety present in current literature.

Chapter 3 contains detailed technical explanations of the deep learning methods used throughout this thesis. It details both the mathematical formulations of the models and analyzes the core differences and similarities between them.

Chapter 4 outlines the dataset utilized for model training and details the step-by-step data preprocessing pipeline. Additionally, it explains the hyperparameter configurations used during model development and defines the mathematical formulas for the chosen evaluation metrics.

Chapter 5 provides a detailed summary of the hyperparameter optimization results for each of the four models, categorized by model architecture, evaluation scores, and hyperparameter selections. The Mean Absolute Error (MAE) score is used as the baseline metric to identify the best hyperparameter configuration for each model, resulting in four top-contending architectures.

Chapter 6 concludes the thesis by discussing the final implications of the experimental results and proposing concrete directions for future research and model improvements.

2 Literature Review

2.1 Machine Learning methods

As stated by Zhang et al. [3], machine learning methods can be divided into two categories: unsupervised and supervised learning. Unsupervised learning methods include techniques such as clustering and dimensionality reduction. Zhang et al. [3] name multiple supervised learning methods, such as k-Nearest Neighbors (kNN), Support Vector Machines (SVM), Back-Propagation Neural Networks (BPNN), Artificial Neural Networks (ANN), and Extreme Learning Machines (ELM). Li et al. [2] mention machine learning-based methods such as the Kalman Filter (KF), Support Vector Regression (SVR), Gaussian Process Regression (GPR), Random Forest (RF), Linear Regression Model (LRM), Autoregressive Model (AR), single-point neighbor search, an improved cultural particle swarm method, a second-order rational Bézier curve coefficients estimation method, Bayesian Networks, an improved beetle antennae search algorithm, an image processing method, an agent-based simulation model, and the Korea Operational Oceanographic System (KOOS). Li et al. [4] mention KF, SVR, GPR, BPNN, and RF methods as well.

An example of an unsupervised learning method mentioned by Zhang et al. [3] is based on Principal Component Analysis (PCA). PCA is applied to each trajectory to generate feature vectors. These vectors are then processed by a Gaussian Mixture Model (GMM) clustering algorithm. Then, each cluster obtained is a likely future

route for the ship to follow. Another example of an unsupervised learning method mentioned is based on Next-Point Connection (NPC) clustering. This method uses a 3-dimensional space to represent trajectory data, which includes time difference, error distance and includes space-time angle. This is used to make the search scope for the next data point more narrow.

For supervised learning, the kNN method has been used in a variety of ways for ship trajectory prediction, as mentioned by Zhang et al. [3]. It has been used as a regression task where traveling duration is set by the median value of the k nearest neighbors. To address multiclass classification problems, the kNN method has also been used as a classifier where each predetermined coordinate is represented as a square cell.

Zhang et al. [3] mention a regression model based on the Least Squares SVM (LS-SVM) method. In it, parameters are optimized by utilizing Particle Swarm Optimization (PSO). Another study also introduced an Adaptive Chaos Differential Evolution (ACDE) method for parameter optimization in SVM-based models. A typical SVM model is only able to handle a single-dimensional output.

Two variants of ANNs are mentioned by Zhang et al. [3]. One utilizes k-means clustering to group historical trajectories of ships into separate groups, which is followed by assigning a label to each group to train an ANN model. The other method implements the Levenberg-Marquardt algorithm, which updates the parameters while training an ANN model. By introducing the Hessian matrix, the Levenberg-Marquardt algorithm is able to learn the network's parameters. It is notably fast at converging on lower-dimensional problems.

More ANN variants mentioned by Li et al. [2] are the Generalized Regression Neural Network (GRNN), ELM, a neuroevolution ANN, and the Multi-Layer Perceptron (MLP). These have the weakness of high computational cost and spatial complexity due to a high number of parameters.

Li et al. [4] mention BPNN as a multilayer feedforward network variant of ANN. It is trained by the error backpropagation algorithm, and with good training and network design, it is able to be very accurate at predictions. Since BPNN is able to model nonlinear relationships, it is very fitting for capturing complex trajectory patterns. It is able to learn patterns and features present in the data automatically during training. BPNN's effectiveness is highly dependent on the data quality, which can make training time and computing resource requirements very high for large-scale datasets. BPNNs have limitations when used for long-term prediction and complex dynamic trajectories.

A BPNN equipped with strong non-linear approximation ability is mentioned by Zhang et al. [3]. It has the weakness of getting trapped in local minima because of the limitations of the gradient descent algorithm.

Zhang et al. [3] describe ELM as a fast and robust algorithm. It is a feed-forward network that is notable for its fast training speed, requiring only few parameters and good generalization ability. A variant of ELM that is based on multiple ELMs is also mentioned. As stated by Zhang et al. [3], a single ELM is performed on each cluster to train the model, and then a weighted kNN method is used to select different trained models for forecasting.

Li et al. [2] mention that the LRM model is often used for time series prediction. Because LRM is easy to overfit, it is best suited for short-term lineear ship trajectory prediction.

Burger et al. [5] states that the KF method is commonly used for prediction. The KF method is a state estimation algorithm that combines a priori experience with rule updates. In ship trajectory prediction, the KF method is used to predict the future track using velocity information alongside historical position. The KF method is based on a observation model and linear dynamic model. It continuously fuses measurement data and predictive models to estimate the current and future

state of the ship. It is able to handle noise and uncertainty during forecasting. It excels in handling ship trajectories with nonlinear relationships. It is able to use recursion to perform real-time dynamic state estimation while updating trajectory information.

The KF method mentioned by Li et al. [2] is able to use observation data to estimate the state of moving targets and predict the next timestamp's trajectory point. The KF method is usually only used for linear systems, which is why there have been proposals of improved methods based on the KF model, such as the Extended Kalman Filter (EKF). The KF method extensions do tend to share difficulties in long-term prediction and route planning due to the high error rates in long-term prediction. Other KF method extensions mentioned are a combination of weighted fast marching square algorithm and the KF method for path planning of unmanned surface ships, an integration of video image processing and the KF algorithm, and a Discrete Kalman Filter (DKF).

As mentioned by Perera et al. [6], the Extended Kalman Filter (EKF) is able to estimate the position, speed, and accelerations of ships. The major dervation between EKF and KF is EKF's nonlinearity.

Gao et al. [7] states that SVR methods have been commonly used in ship trajectory prediction and are able to outperform linear regression. By constructing a hyperplane in a feature space, SVR is able to maximize the margin between observed and predicted values. SVR is able to utilize kernel functions to map data to higher-dimensional feature spaces. This makes SVR effective in handling nonlinear relationships and makes the model more flexible. For training, relevant ship states and environment are selected. For tuning, suitable kernel functions and parameters are selected. A trained SVR model is able to predict ship's heading, speed and position for a specific time period. Possible SVR kernel function types are linear, polynomial, and radial basis function. SVR has a good generalization ability and is

able to handle noisy and high-dimensional datasets.

Li et al. [2] mention the SVR model as a combination of SVM and regression and some of its variants. Three improved SVR models mentioned are an SVR optimized by Dimension Learning Grey Wolf Optimizer (DLGWO-SVR), by Adaptive Chaotic Differential Evolution (ACDE-SVR), and an Online and Multiple Outputs Least-Squares Support Vector Regression based on Selection Mechanism (SM-OM LSSVR). SVR models have issues of parameter selection being challenging and high computational cost.

The GPR model mentioned by Li et al. [2] is an improvement of Bayesian Linear Regression. GPR model variants mentioned include a highly data-dependent sparse GPR model to aid intelligent ship navigation and a nonparametric GPR-based Bayesian prediction model to predict trajectory while explaining uncertain lateral movement, which unfortunately is not able to handle high-dimensional data. A model that is a combination of LRM and GPR is also mentioned by Li et al. [2]. It is able to predict the destination and the trajectory of a ship during navigation.

As mentioned by Li et al. [4], GPR is a random process. It consists of an infinite number of Gaussian random variables that are defined in a spatial or continuous temporal domain. GRP's ability to model noise makes it useful in ship trajectory prediction in regards to risk assessment and decision making. GPR is able to handle nonlinear relationships and characteristics in ship trajectory prediction alongside complex dynamic characteristics. It is also suitable for prediction even for small datasets. The disadvantages of GPR include high computational complexity, which makes it suboptimal for large datasets. This makes hyperparameter optimization crucial for GPR.

Li et al. [2] mention that RF is a comprehensive algorithm with decision trees. RF has a many useful applications for ship trajectory prediction, such as the prediction of arrival points, destination points, ship speed, and trajectory prediction. RF

algorithms are also prone to overfitting.

2.2 Deep learning methods

According to Li et al. [4] and Zhang et al. [3], deep learning methods started to become common in ship trajectory prediction since late 2019. For example, in publications released in 2023, two were based on machine learning methods and six based on deep learning methods. The most common deep learning methods used after 2020 are Long Short-Term Memory (LSTM), Sequence-to-Sequence (Seq2seq), and Bi-directional Long Short-Term Memory (Bi-LSTM), as stated by Li et al. [4].

Many deep learning methods based on Recurrent Neural Networks (RNNs) are commonly used. Gated Recurrent Unit (GRU), Bi-directional Gated Recurrent Unit (Bi-GRU), and Transformer are also used in ship trajectory prediction, according to Li et al. [2]. Zhang et al. [3] mention methods such as LSTM, encoder-decoder architecture, GRU, Convolutional Neural Network (CNN), Transformer, Generative Adversarial Network (GAN), Fully-Connected Neural Network (FCNN), and Deep Neural Network (DNN). Yang et al. [1] also mention methods such as RNN and its extensions LSTM and GRU.

Kim and Lee [8] mention a hybrid deep learning framework based on CNN and FCNN, which is able to learn movement patterns of all ships in a harbor area. CNN extracts hidden features from ship trajectory data with attributes of speed, position, and course. FCNN obtains higher-level features from other attributes such as Estimated Time of Arrival, ship length, destination, and ship type.

Zhang et al. [3] mention a GAN framework that was first proposed by Wang and He [9], which is a deep generative model. It can be used to forecast ship trajectories. GAN consists of two individual networks including a discriminator network and a generator network.

The generator network captures data distribution. Whether sample is real or

generated is determined by the discriminator network. In an example of GAN given, the generator network was formed by an encoder-decoder LSTM with interaction and attention mechanisms. The discriminator network was a regular LSTM network. GAN performed considerably better compared to Seq2seq and the Kalman Filter (KF) models.

Chen et al. [10] mention a framework based on DNN that was developed for predicting trajectories of general-purpose merchant vessels. DNN methods are known to suffer from overfitting.

The Seq2seq model is mentioned by Capobianco et al. [11]. It belongs to one of the encoding-decoding structures. Mapping the input to the output sequence is possible even with unequal lengths. Seq2seq variants mentioned are a combination of RNN and Seq2seq, a Variational Recurrent Autoencoder decoder and density-based clustering algorithm, and an optimized Seq2seq model by spatio-temporal features. Seq2seq models are suited for short-term trajectory prediction over long-term trajectory prediction, in which the performance is poor.

Li et al. [4] mention the Spatial Temporal Graph Convolutional Network (STGCN) model. It is good at handling spatiotemporal data. In ship trajectory prediction, STGCN is able to accurately predict future trajectories by capturing spatio-temporal dependencies in data. Ship trajectory data is viewed as a spatio-temporal graph with nodes that correspond to ship positions and features, and edges between nodes that indicate spatio-temporal relationships. The edges can be representations of adjacency relationships between ships or connections between ships at different time points. STGCN performs convolutions on spatiotemporal graphs and includes spatiotemporal information into the network. The input during training is the combination of historical trajectory data and future trajectory information, after which supervised learning is used to train the network. STGCN is very adept at exploiting the spatio-temporal structure of ship trajectory data. This allows the capture of

associations and dependencies between ships. STGCN is able to handle large-scale datasets and spatio-temporal series data of variable lengths. It is also able to utilize multiple features such as speed and ship type to improve prediction accuracy. The disadvantage of STGCN is high computational complexity.

2.2.1 Long Short-Term Memory

According to Li et al. [4] and Zhang et al. [3], LSTM is one of the most common deep learning methods used for ship trajectory prediction. As mentioned by Li et al. [2], there are many integrated models of LSTM, such as a Trajectory-based Similarity Search Prediction Model (TSSPL), a Context-Aware LSTM (C-LSTM), a federated deep learning-based method (Conv LSTM), an Accumulated Long Short-Term Memory model (ALSTM), a Multi-step Prediction LSTM (MP-LSTM), and a Bidirectional Recurrent Mixture Density Network (Bi-RMDN). LSTM models solve the issue of RNN's short-term memory by combining short-term and long-term memory through dedicated gate controls. This solves the problem of vanishing gradients. LSTM is stated by Yang et al. [1] to be the best choice for trajectory prediction as it is highly suitable for learning data with long-term dependence. Zhang et al. [3] mention a study where a single LSTM model was trained for each ship. This context-aware LSTM outperformed a general LSTM model in regards to convergence speed, oscillation amplitude, and forecasting accuracy.

Other variants of LSTM include the LSTM encoder-decoder architecture, as mentioned by Zhang et al. [3]. One example of an encoder-decoder LSTM was built as a sub-framework within GAN, and it exhibited superior results compared to the OU process model. Another example of an encoder-decoder LSTM featured a study where the Mediterranean Sea was depicted as a series of non-overlapping grids, with each grid being assigned a unique code. An encoder-decoder LSTM was built by taking an input of a sequence of grid codes that a ship had passed

through previously. Each waypoint of the trajectory of a ship is represented by a certain grid cell area. The outputs are potential grids that the ship could travel through. A third example of an encoder-decoder LSTM was used to predict upper and lower geographical bounds. It consisted of two networks, which were trained by a joint supervisor mechanism. Another example of an encoder-decoder LSTM framework consisted mainly of feature extraction and model learning. To extract features from trajectory data, an encoder-decoder LSTM framework was used. As stated by Zhang et al. [3], the extracted features, alongside geographical information of ships, were concatenated as the input of an attention-based Bi-LSTM for regression model training. An attention-based encoder-decoder LSTM architecture is also mentioned. This was made for autonomous vessel prediction. It incorporated both spatial and temporal attention mechanisms, which allow different information to be focused on. It outperformed regular LSTM and LSTM with only one attention mechanism.

2.2.2 Bi-Directional Long Short-Term Memory

Bi-LSTM is one of the most common deep learning methods and LSTM variants used in ship trajectory prediction, according to Li et al. [4], and it tends to perform better compared to standard LSTM models. Bi-LSTM has the ability to handle data from both the past and future [2]. As mentioned by Zhang et al. [3], the bidirectional structure allows Bi-LSTM to enhance the relevance between historical and future time-series data.

Li et al. [2] mention several hybrid models based on Bi-LSTM. In these models, Bi-LSTM is combined with data denoising, spectral clustering, attention mechanisms, or combined convolution layers alongside attention mechanisms and dense layers. Other examples of Bi-LSTM hybrid variants are the Bidirectional Long Short-Term Memory-Recursive Neural Network (BLSTM-RNN) and a Dual-pass

Long Short-Term Memory network model.

2.2.3 Gated Recurrent Unit

Wang et al. [12] mention the GRU variant Bi-directional GRU, where GRU is an LSTM variant with fewer trainable parameters, and an encoder-decoder architecture based on GRU blocks. These have shown promise in being more suitable for short-term trajectory prediction tasks compared to standard LSTM and GRU.

Li et al. [4] also mention how GRU is more suited toward short-term trajectory prediction compared to LSTM.

Li et al. [2] mention a GRU variant that solved the gradient dispersion problem: a prediction model based on GRU and a Multi-Scale Convolutional Neural Network (MSCNN). MSCNN is used to extract spatial-temporal characteristics and is combined with GRU, an attention mechanism, and an autoregressive model to predict ship trajectories.

2.2.4 Temporal Convolutional Network

Zhang et al. [3] mention the Temporal Convolutional Network (TCN) as a promising technique. TCN's advantage is the usage of dilated causal convolutions. These dilated causal convolutions are used to ensure that during the modeling of temporal data the temporal order is retained. Dilated convolutions are able to enable an exponential increase of the receptive field.

2.2.5 Transformer

Li et al. [4] mention how the Transformer model is capable of predicting future ship positions and movements by learning patterns from historical data. It is able to effectively capture contextual information alongside dependency relationships from different timesteps. This helps it to predict future ship trajectories accurately. Data

points such as position coordinates, timestamps, and historical trajectory attributes such as speed and ship type are used as inputs for the Transformer model. It uses a self-attention mechanism and an encoder-decoder structure to learn spatiotemporal relationships to predict future trajectory points. Using real future trajectories as the target, the model is able to be trained with unsupervised learning. The model parameters can be optimized for accuracy by minimizing the difference between predicted and actual trajectories. The Transformer model has the disadvantages of requiring many parameters and high computational resources. It also struggles with long-term dependencies, has limitations on sequence length, and requires large datasets to be effective.

Zhang et al. [3] mention a Transformer-based neural network that embeds sequential inputs such as longitude, latitude, COG, and SOG into high-dimensional discrete vectors. The model is then used as a classifier to predict the probability distribution for each attribute over discrete bins.

Li et al. [2] mention a couple of Transformer-based methods. One of these is the TripConvTransformer model. It consists of global, local, and trend convolutions. It integrates meteorological data to a simplified Transformer architecture. K-means methods were used for discretization analysis of meteorological data, but this was found to be ineffective for capturing the features of meteorological data.

Another Transformer-based method mentioned by Li et al. [2] is the TrAISformer model. It was able to enhance prediction accuracy by incorporating long-term correlations of AIS data using a probabilistic Transformer structure.

2.3 Other methods

There are also other methods for ship trajectory prediction that do not strictly fall under deep learning or machine learning-based methods. As stated by Zhang et al. [3], these include methods such as simulation methods, statistical methods, and

hybrid methods. Simulation methods work by modeling the process of creating a digital prototype of a physical model. Motion models and Constant Velocity Models (CVM) are examples of simulation models. Simulation models are rarely used on their own anymore, but are present in many hybrid models.

Zhang et al. [3] mention several hybrid methods. The bases of these hybrid methods include simulation methods, statistical methods, machine learning methods, and deep learning methods. An example of a hybrid model mentioned by Zhang et al. [3] is a model composed of clustering, a machine learning method, and MTEM. By not utilizing the criterion of nearest neighbors, points are clustered within a distance based on the Vincenty Distance metric. The clustered AIS data is then further clustered by SOG and COG. To ensure that only one point from each ship is included in the clustered region, duplicate points are removed. With averaged COG and SOG of the clustered data, a prediction algorithm is performed.

Zhang et al. [3] mention several statistical methods. The Single Point Neighbor Search (SPNS) predicts ship trajectories by estimating the course and speed of its close neighbors. A weakness of SPNS is its sensitivity to parameter settings. The Multiple Trajectory Extraction Method (MTEM) is an improved version of SPNS.

A multi-step hybrid model involving simulation, statistical, and deep learning methods is mentioned by Gao et al. [13]. The three steps of the model include support point prediction, destination point prediction, and trajectory interpolation. A deep learning model such as LSTM is trained to predict a support point. Destination point prediction is based on historical data filtering. This step requires that two trajectories satisfy several predetermined conditions. The trajectory is simulated by the cubic spline interpolation technique after the information on the support point and destination is derived.

The Neighbor Course Distribution Method (NCDM) is helped by GMM to generate multiple possible trajectories while being able to measure uncertainty and

handle multimodality [3].

Similarity search method models locate the most similar trajectories from historical data to convolve and aggregate them to form a temporal probability density field. Similarity search models display high accuracy in ships' position prediction [3].

Alizadeh et al. [14] mention a similarity search model based on LSTM. To search for the most similar trajectory, the DTW method is used. Then, to predict spatial distances, the LSTM model is built based on the historical AIS data. This is used to predict movement trends of a ship.

The Monte Carlo Method (MCM) simulates a large number of synthetic trajectories to generate the probability field, as mentioned by Scheepens et al. [15].

A k-order multivariate Markov chain model is able to use location, speed, and course to build a state-transition matrix for long-term ship trajectory prediction. It is a grid-based method where the ocean is divided into overlapping grids, and the model is used to predict which grid cell the ship is in [3].

A hybrid framework for trajectory prediction consisting of three modules is mentioned by Murray and Perera [16]. The modules are trajectory clustering, trajectory classification and trajectory prediction. GMM clustering algorithm is used for the trajectory clustering module to group similar historical trajectories. To classify selected trajectories into clusters a kNN is used for the trajectory classification module. For trajectory prediction module a dual linear autoencoder is used on the selected cluster trajectories.

Suo et al. [17] mentions a hybrid model based on the combination of unsupervised clustering such as DBSCAN and deep learning method such as GRU. A DBSCAN algorithm is applies to AIS trajectory data to obtain the main ship trajectory zone. This is done to eliminate redudant data by performing similarity analysis before training a GRU model.

A hybrid framework for trajectory prediction utilizing Multi-Layer Neural Network (MLNN) is mentioned by Xiao et al. [18]. MLNN is used to predict COG and SOC. Based on the derived COG and SOC motion modeling is applied to obtain the positions of the ships. If a movement pattern is found on the historical knowledge base the PF method is used to correct the COG sequence.

Sang et al. [19] mentions a hybrid model that is the combination of Exponential Smoothing Method (ESM) and motion model for ship position prediction. ESM is used to forecast COG and SOC. By including the predicted COG and SOC into a simple motion model alongside other attributes like acceleration and ROT position of the ship is predicted. A similar method that uses Expetation Maximation (EM) clustering and trajectory matching is also mentioned. EM clustering and trajectory matching is used to calculate COG and SOC. Afterwards a motion model is used for future trajectory estimation.

A knowledge-based model for position prediction is mentioned by Mazzarella et al. [20]. A kNN classifier is used by the model to match one possible route from the knowledge base to the future route. In a case where a suitable match is not found a simple CVM model is used for prediction. If a matching route is found a knowledge-based velocity or knowledge-based PF method is used for ship trajectory prediction. The knowledge-based PF method is more effective than the knowledge-based velocity method.

Li et al. [21] mentions a hybrid model for long-term ship prediction featuring clustering and deep learning. To discover regional clusters based on positional data such as longitude and latitude of the AIS data DBSCAN clustering is utilized. Then based on the regional clustering a regional LSTM model is built for separate training and prediction.

A hybrid method based on encoder-decoder LSTM network and wild bootstrapping is mentioned by Venskus et al. [22]. Encoder-decoder LSTM network is used to

provide geographical position distribution outputs for wild bootstrapping technique to learn.

Murray and Perera [23] mentions a hybrid framework based on clustering and deep learning. The framework is composed of three modules which are the clustering module, the classification module and the local behavior module. Variational encoder-decoder structure is used in the clustering module to extract latent representations of each trajectory. The output is then fed into Hierarchical DBSCAN (HDBSCAN). To assign multiple clusters to a new trajectory a Bi-GRU model is trained in the classification model. Bi-GRU is then used as a base for local models for each cluster in the local behavior module. To perform a cluster-wise prediction the resulting clusters from classification module are utilized.

A hybrid machine learning based model mentioned by Li et al. [2] combines a physical model, Modular Logical Neural networks (MLNN) and PF. It is able to predict risk assessments and collision detections reliably.

Another hybrid machine learning based method is mentioned by Li et al. [2]. A Multi-Output Hybrid Predictor (MHP) method is utilized in ship collision avoidance and autonomous-ship decision making.

Li et al. [2] mentions another hybrid machine learning based prediction model based on the combination of Model Predictive Controller (MPC) and Radial Basis Function (RBF). It is utilized in collision avoidance and multi-ship moving control.

3 Background

3.1 Automatic Identification System

The Automatic Identification System (AIS) is a system designed for sending information on the location, speed, course, and identity of a ship at frequent time intervals. AIS is obligated to be used by all passenger ships, tankers, and other ships over 300 tons during international voyages. [24] AIS constantly sends both static and variable information to nearby ships and onshore authorities wirelessly. Static information includes details such as the Maritime Mobile Service Identity (MMSI), length, and width. Dynamic information includes, for example, ship position by latitude and longitude, Speed Over Ground (SOG), and Course Over Ground (COG). [2]

AIS uses two Very High Frequency (VHF) channels for communication. These channels are AIS1 (channel 87B, 161.975 MHz) and AIS2 (channel 88B, 162.025 MHz). This allows AIS receivers to be either dual- or single-channel. A dual-channel AIS receiver, compared to a single-channel one, is able to display more information, complete messages, and update information. An AIS receiver assembly consists of the receiver module, a VHF antenna with cables, a laptop with software able to record and interpret message streams, and, in cases where the laptop does not power the assembly, a separate power source. [25]

3.2 Maritime Situational Awareness

Maritime Situational Awareness (MSA) is a multifaceted concept that plays a critical role in maritime safety and security. An important aspect of MSA is to anticipate future implications by involving the understanding of a ship's surroundings and interpreting current information. The best ways to enable the prevention of accidents and the ability to respond to unusual or threatening situations involve proactive management of maritime risks alongside real-time monitoring and analysis. MSA is usually divided into three levels: Level 1, consisting of perception; Level 2, consisting of interpretation; and Level 3, consisting of anticipation.

Level 1 consists of recognizing surroundings and risks by perceiving the attributes, status, and dynamics of relevant elements in the maritime environment. Key information, such as the speed and position of the ship, locations of nearby vessels, and environmental factors like sea conditions, is provided by onboard sensors and needs to be monitored constantly. The information gathered is categorized further into groups such as ship status, route information, traffic, and weather conditions.

Level 2 involves assessing safety significance by building on Level 1 through synthesizing and interpreting the gathered data. This enables the understanding of navigational goals and objectives by integrating diverse data points through pattern recognition, evaluation, and contextual understanding. The gathered data is compared with the ideal system state, taking into account the effects of external factors on navigation.

Level 3 consists of predicting future conditions and risks based on current data. It includes the projection of a ship's future position, predicting nearby vessels' movements, and assessing future weather conditions. Level 3 requires both the previous Level 1 and Level 2 to predict and prepare for future operational states, such as predicting nearby ship traffic and detecting abnormal activities like illegal behavior

or navigational hazards.

Maintaining MSA is the responsibility of the Officer of the Watch (OOW). This requires constant monitoring of the maritime environment, traffic, sea depth, and weather conditions. The OOW needs to be knowledgeable about the ship's characteristics, such as construction, propulsion systems, and onboard equipment, to make informed decisions regarding the identification and prevention of potential risks. The OOW is also responsible for tracking the ship's position alongside preparing for unexpected events and emergencies. This is extremely challenging due to demanding conditions and often leads to human errors regarding MSA. This is why developing AI-based systems and solutions to automate monitoring tasks, such as ship trajectory prediction and detecting illegal behavior, is critical. AI systems providing Level 1 situational awareness have been especially highlighted recently. [26]

3.3 Ship Trajectory Prediction

According to Zhang et al. [3], a ship trajectory can be expressed as a sequence of tuples $\{x_t, t \in T\}$, where $x_t = (l_t, o_t, \dots)$ consists of geolocations l_t and other attributes o_t such as speed, heading, or ship type. Here, T is a set of timestamps defined as $T = \{1, 2, 3, \dots\}$.

Ship trajectory prediction has become critical to the safety of maritime transportation. The volume of international trade has increased, and maritime transportation plays a large role in supporting global trade. Approximately 90% of the global trade is handled by maritime transportation. Vessel trajectory research is typically organized into three levels: trajectory data, trajectory pattern and model, and domain applications. [3] Due to the widespread access to AIS data over the last decade, the most common prediction methods for ship trajectories are based on historical data. [2]

According to Zhang et al. [3], a vessel trajectory prediction task can be expressed

at future timestamps $t+1, t+2, \ldots, t+a$ as $Y = \{x_{t+1}, x_{t+2}, \ldots, x_{t+a}\}$, where a denotes the prediction horizon, given a fully observed trajectory over history timestamps $\{1, 2, \ldots, t\}$ denoted as $X = \{x_1, x_2, \ldots, x_t\}$.

As AIS has been widely adopted, a massive amount of data is generated, which can reach up to 1 trillion pieces of information per day. Due to the large volume of AIS data, researchers are highly interested in exploring different methods to utilize it. Consequently, outlier detection and filtering have become critical aspects of utilizing AIS data due to the inherent noise and inaccuracies present in raw transmissions. [1]

3.4 Deep Learning

Advantages of using deep learning methods are the ability to analyze complex relationships and automatic feature identification. Deep learning methods are also suitable for large datasets and have good fitting ability for complex trajectories. Weaknesses of deep learning methods include the chance of overfitting with smaller datasets, being hard to understand, and requiring a lot of computing power. [4]

3.5 Recurrent Neural Network

Recurrent Neural Networks (RNNs) are used for extracting long-term dependencies in sequential data. An RNN's unique memory unit allows it to perform short sequence predictions. Due to the sequence length being unknown in practical applications, RNNs are limited by gradient vanishing and gradient explosion during the learning process. RNNs have many variants such as Long Short-Term Memory, Temporal Convolutional Network, and Gated Recurrent Unit. [1] A simple RNN

internal memory unit h_t can be expressed as follows, as stated by Yang et al. [1]:

$$h_t = f(Wx_t + U_f h_{t-1} + b)$$

with f as the activation function, U_f and W as weight matrices of the hidden layer, b as the bias, and x_t as the input vector.

3.6 Bidirectional Long-Term-Short-Term Memory

3.6.1 Long-Term-Short-Term Memory

Long Short-Term Memory (LSTM) is an RNN architecture proposed by Sepp Hochreiter and Jürgen Schmidhuber.

It is designed to handle long sequences in its short-term memory.

This is made possible by enforcing constant error flow through the internal states of special units of a gradient-based algorithm. [27] LSTM is a sequence-to-sequence prediction model that is both scalable and effective. It is considered one of the best choices for track prediction thanks to its good feature selection sequence and the lag period between the input sequence and output prediction. LSTM is highly suitable for learning data with long-term dependencies. [1] However, LSTM has the weakness of inferior and time-consuming parallel processing.

LSTM features three types of gates: input, forget, and output gates. The removal of unnecessary information from the cell state is determined by the first layer of the memory gate. It can be expressed as follows, as stated by Yang et al. [1]:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

where f_t is the forget gate activation vector at time t, W_f and U_f are the weight matrices, σ is the sigmoid activation function, x_t is the input vector, h_{t-1} is the

hidden state vector at time $t-1$, and b_f is the bias vector.

The information from the input vector that is stored in the cell state is determined by the input gate. The value is updated by the decision i_t, and a tanh layer creates the new candidate state value of $\tilde{c}_t$. These can be expressed as follows, as stated by Yang et al. [1]:

$$i_t = \sigma(W_i \cdot x_t + U_i \cdot h_{t-1} + b_i)$$

$$\tilde{c}_t = \tanh(W_c \cdot x_t + U_c \cdot h_{t-1} + b_c)$$

where the input gate activation at time t is expressed by i_t. Weights are represented by W_i, U_i, W_c, and U_c. Bias terms are b_i and b_c. The cell state c_t being updated at time t can be expressed as follows, as stated by Yang et al. [1]:

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

Output information in the current timestep is produced by the third layer. It can be expressed as follows, as stated by Yang et al. [1]:

$$o_t = \sigma(W_o \cdot x_t + U_o \cdot h_{t-1} + b_o)$$

The output gate activation at time t is denoted by o_t. Weights are represented by W_o and U_o. The bias term is represented by b_o.

The output value of the cell is expressed as follows, as stated by [1]:

$$h_t = o_t \odot \tanh(c_t)$$

The output gate at time t is denoted by o_t. The activation function is tanh, and c_t is the state of the cell at time t. The effective information is output, and invalid

information is forgotten after the data passes through the three gates.

3.6.2 Bidirectional Long-Term-Short-Term Memory

The Bidirectional Long Short-Term Memory (Bi-LSTM) model is composed of two independent LSTM models. [4] Unlike a standard LSTM, a Bi-LSTM is able to store both past and future data. A Bi-LSTM processes information in two directions: from past to future and from future to past. [2] This helps with accuracy by enhancing the relevance of past and future data. [3] A weakness of Bi-LSTM is the inability to be computed in parallel. Bi-LSTM also suffers from the information at the beginning of a long sequence not being well conveyed. [4]

3.7 Temporal Convolutional Network

Temporal Convolutional Networks (TCN) are a class of time-series models that are able to capture long-range patterns using a hierarchy of temporal convolutional filters. [28] TCN has its own advantages over other deep learning methods. It has the ability to utilize dilated causal convolutions. Causal convolutions are able to guarantee that the temporal order is not violated when modeling temporal data. Dilated convolutions are able to enable exponential expansion of the receptive field while not sacrificing coverage. TCN excels in capturing the spatio-temporal dependencies among data. [3]

Lea et al. [28] presented two types of TCNs: the Encoder-Decoder TCN (ED-TCN) and the Dilated TCN. The ED-TCN only uses a hierarchy of temporal convolutions, pooling, and upsampling. It is able to capture long-range temporal patterns efficiently. The ED-TCN has a small number of layers, each containing a set of long convolutional filters. The Dilated TCN uses dilated convolutions instead of pooling and upsampling while adding skip connections between layers. The Dilated TCN

has more layers. On a small number of time steps each layer uses dilated filters to operate.

The Dilated TCN is adapted from the WaveNet model. For the Dilated TCN, a series of blocks is defined. Each block contains a sequence of L convolutional layers. The activations in the j-th block and the l-th layer are given by $S^{(j,l)} \in \mathbb{R}^{F_w \times T}$. The first block is defined by the input data. The input for the following block $S^{(j,l)}$ is the output of the previous block $S^{(j-1,L)}$. Every layer has the same number of filters F_w. By using skip connections the model is able to combine activation from different layers. Every layer consists of a set of dilated convolutions with a rate parameter s, a non-linear activation $f(\cdot)$, and a residual connection that combines the convolutional signal and the layer's input. Convolutions are only applied over two time steps, t and $t - s$. The filters are parameterized by $W = \{W^{(1)}, W^{(2)}\}$, where $W^{(i)} \in \mathbb{R}^{F_w \times F_w}$. The bias vector is $b \in \mathbb{R}^{F_w}$. The result of the dilated convolution at time t is $\hat{S}_t^{(j,l)}$, and $S_t^{(j,l)}$ is the result of adding the residual connection, which can be expressed as follows, as stated by Lea et al. [28]:

$$\hat{S}_t^{(j,l)} = f(W^{(1)} S_{t-s}^{(j,l-1)} + W^{(2)} S_t^{(j,l-1)} + b)$$

$$S_t^{(j,l)} = S_t^{(j,l-1)} + V \hat{S}_t^{(j,l)} + e$$

where $V \in \mathbb{R}^{F_w \times F_w}$ and $e \in \mathbb{R}^{F_w}$ are set as the weights and biases for the residual. The parameters $\{W, b, V, e\}$ are separate for each layer.

The dilation rate increases for consecutive layers within a block so that $s_l = 2^l$ which his enables an increase of the receptive field even with a low number of parameters. The output for each block is summed using skip connections with $Z^{(0)} \in \mathbb{R}^{F_w \times T}$. This can be expressed as follows, as stated by Lea et al. [28]:

$$Z_t^{(0)} = \text{ReLU}\left(\sum_{j=1}^{B} S_t^{(j,L)}\right)$$

The set of latent states is defined as $Z_t^{(1)} = \text{ReLU}(V_r Z_t^{(0)} + e_r)$ for bias e_r and the weight matrix $V_r \in \mathbb{R}^{F_w \times F_w}$. The predictions given for each time t are expressed as follows, as stated by Lea et al. [28]:

$$\hat{Y}_t = \text{SoftMax}(U Z_t^{(1)} + c)$$

for bias $c \in \mathbb{R}^C$ and weight matrix $U \in \mathbb{R}^{C \times F_w}$.

As the filters in the Dilated TCN are smaller than in the ED-TCN, more layers and blocks are needed to make the receptive fields equal. For the number of blocks B and the layers per block L, the receptive field is of length $r(B, L) = B \cdot 2^L$.

3.8 Gated Recurrent Unit

The Gated Recurrent Unit (GRU) was proposed by Cho et al. [29]. GRU models have a high calculation cost, but they have addressed the shortcomings of LSTM. [2] Advantages of using a GRU are low amount of parameters, lower risk of overfitting and faster converge speed.

GRU has the weakness of expression ability decreasing if there is too much data. The time complexity of GRU methods is $O(n^2)$. [4] Like the LSTM unit, the GRU has gating units that modulate the flow of information inside the unit. A clear difference between LSTM and GRU is the GRU's lack of separate memory cells. At time t, the activation h_t^j is a linear interpolation between the previous activation h_{t-1}^j and the candidate activation $\tilde{h}_t^j$, which can be expressed as follows, as stated by Chung et al. [30]:

$$h_t^j = (1 - z_t^j) h_{t-1}^j + z_t^j \tilde{h}_t^j$$

The update gate z_t^j decides by how much the unit updates its content or activation. The update gate can be expressed as follows:

$$z_t^j = \sigma(W_z x_t + U_z h_{t-1})^j$$

The GRU takes a linear sum between the existing state and the newly computed state. Without any control mechanism GRU is able to expose the whole state each time which is another difference between LSTM and GRU. The candidate activation $\tilde{h}_t^j$ can be expressed as follows, as stated by Chung et al. [30]:

$$\tilde{h}_t^j = \tanh(W x_t + U(r_t \odot h_{t-1}))^j$$

The set of reset gates is r_t, and $\odot$ is an element-wise multiplication. When r_t^j is close to 0, the reset gate makes the unit act as if it is reading the first symbol of an input sequence, essentially forgetting the previously computed state.

The reset gate can be expressed as follows, as stated by Chung et al. [30]:

$$r_t^j = \sigma(W_r x_t + U_r h_{t-1})^j$$

and is computed similarly to the update gate.

3.9 Temporal Fusion Transformer

The Temporal Fusion Transformer (TFT) was proposed by Lim et al. [31]. TFT is an attention-based Deep Neural Network (DNN) architecture for multi-horizon forecasting. The prediction of variables of interest at multiple future timesteps is called multi-horizon forecasting. Multi-horizon forecasts allow users to estimate across the entire path, unlike one-step-ahead predictions.

To encode context vectors for other parts in the network TFT incorporates static covariate encoders. Gating mechanisms alongside sample-dependent variable selection is used to minimize the effect of irrelevant inputs. Processing of known and

observed input locally is handled by a sequnce-to-sequence layer. Long-term dependencies in dataset are learned by a temporal self-attention decoder.

The TFT architecture can be described with 4 constituents: gating mechanisms, variable selection networks, static covariate encoders, and temporal processing and prediction intervals. [31]

The Gated Residual Network (GRN) is a building block of TFT. The GRN takes in a primary input a together with an optional context vector c. GRN_ω can be expressed as follows, as stated by Lim et al. [31]:

$$\text{GRN}_\omega(a, c) = \text{LayerNorm}(a + \text{GLU}_\omega(\eta_1))$$

$$\eta_1 = W_{1,\omega}\eta_2 + b_{1,\omega}$$

$$\eta_2 = \text{ELU}(W_{2,\omega}a + W_{3,\omega}c + b_{2,\omega})$$

where ELU is the Exponential Linear Unit activation function. The intermediate layers are $\eta_1 \in \mathbb{R}^{d_{\text{model}}}$ and $\eta_2 \in \mathbb{R}^{d_{\text{model}}}$. LayerNorm is standard layer normalization. The index to denote weight sharing is ω. When $W_{2,\omega}a + W_{3,\omega}c + b_{2,\omega} \gg 0$, the ELU activation acts as an identity function. When $W_{2,\omega}a + W_{3,\omega}c + b_{2,\omega} \ll 0$, the ELU activation generates a constant output, similar to linear layer behavior. [31]

When $\gamma \in \mathbb{R}^{d_{\text{model}}}$ is the input, the GLU can be expressed as follows, as stated by Lim et al. [31]:

$$\text{GLU}_\omega(\gamma) = \sigma(W_{4,\omega}\gamma + b_{4,\omega}) \odot (W_{5,\omega}\gamma + b_{5,\omega})$$

The sigmoid activation function is denoted by $\sigma(\cdot)$. Weights are $W_{(\cdot)} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$. The bias is $b_{(\cdot)} \in \mathbb{R}^{d_{\text{model}}}$. The element-wise Hadamard product is $\odot$, and the hidden state dimension is d_{model}. GLU allows the TFT to control the extent to which the GRN contributes to the original input a. Layers can be skipped over if deemed

necessary. In the cases where there is not context vector the contect input is treated as zero by the GRN. Dropout is applied during training before the gating layer.

TFT is capable of performing variable selection to remove unnecessary noisy inputs and improve performance. Entity embeddings are used for categorical variables and linear transformations are used for continuous variables to transform every input variable into a d_{model}-dimensional vector. The transformed input of the j-th variable at time t is denoted by $\xi_t^{(j)} \in \mathbb{R}^{d_{\text{model}}}$. The flattened vector of all historical inputs can be expressed as follows, as stated by Lim et al. [31]:

$$\Xi_t = \left[\xi_t^{(1)^T}, \ldots, \xi_t^{(m_\chi)^T}\right]^T$$

where variable selection weights are generated by feeding Ξ_t and an external context vector c through a GRN, followed by a Softmax layer. This can be expressed as follows, as stated by Lim et al. [31]:

$$\upsilon_{\chi t} = \text{Softmax}(\text{GRN}_{\upsilon_\chi}(\Xi_t, c_s))$$

The vector of variable selection weights is $\upsilon_{\chi t} \in \mathbb{R}^{m_\chi}$, and c_s is obtained from a static covariate encoder. If the variable is static, the context vector c_s is omitted.

During each timestep, each $\xi_t^{(j)}$ is fed through its own GRN to provide an additional layer of non-linear processing. This can be expressed as follows, as stated by Lim et al. [31]:

$$\tilde{\xi}_t^{(j)} = \text{GRN}_{\tilde{\xi}_{(j)}}(\xi_t^{(j)})$$

The processed feature vector for variable j is $\tilde{\xi}_t^{(j)}$. Every variable has its own $\text{GRN}_{\tilde{\xi}_{(j)}}$, with weights being shared across all timesteps t.

Processed features are combined after they have been weighted by their variable selection weights. This combination can be expressed as follows, as stated by Lim

et al. [31]:

$$\tilde{\xi}_t = \sum_{j=1}^{m_\chi} \upsilon_{\chi t}^{(j)} \tilde{\xi}_t^{(j)}$$

where $\upsilon_{\chi t}^{(j)}$ is the j-th element of vector $\upsilon_{\chi t}$.

Four separate GRN encoders are used to build complex representations of static metadata. Enriching temporal features with static information, local processing of temporal features and producing context for temporal variable selection are the use cases of static variables.

TFT learns long-term relationship between different timesteps by utilizing a self-attention mechanism. The attention mechanism scales values $V \in \mathbb{R}^{N \times d_v}$ based on relationships between keys $K \in \mathbb{R}^{N \times d_{\text{attn}}}$ and queries $Q \in \mathbb{R}^{N \times d_{\text{attn}}}$ that can be expressed as follows, as stated by Lim et al. [31]:

$$\text{Attention}(Q, K, V) = A(Q, K)V$$

where $A(\cdot)$ is a normalization function. [31]

Multi-head attention is used to attend to different representation subspaces. Analyzing attention heads on their own is not indicative of a certain feature's importance, since different values are used in each head. Multi-head attention is modified to share values in each head and to employ additive aggregation of all heads at the output, which can be expressed as follows, as stated by Lim et al. [31]:

$$\text{InterpretableMultiHead}(Q, K, V) = \tilde{H} W_H$$

$$\begin{aligned}
\tilde{H} &= \tilde{A}(Q, K) V W_V \\
&= \left\{ \frac{1}{H} \sum_{h=1}^{H} A(Q W_Q^{(h)}, K W_K^{(h)}) \right\} V W_V \\
&= \frac{1}{H} \sum_{h=1}^{H} \text{Attention}(Q W_Q^{(h)}, K W_K^{(h)}, V W_V)
\end{aligned}$$

Value weights shared across all heads are denoted by $W_V \in \mathbb{R}^{d_{\text{model}} \times d_V}$, and $W_H \in \mathbb{R}^{d_{\text{attn}} \times d_{\text{model}}}$ is used for linear mapping. This allows each head to learn different temporal patterns while having a common set of input features.

The temporal fusion decoder uses a series of layers to learn temporal relationships present in the dataset. These layers are the Locality Enhancement with Sequence-to-Sequence Layer, the Static Enrichment Layer, the Temporal Self-Attention Layer, and the Position-wise Feed-forward Layer. The Locality Enhancement with Sequence-to-Sequence Layer is suited to handle differences between points of significance in relation to their surrounding values, which is useful for cases where there are differing numbers of past and future inputs. The Static Enrichment Layer enhances the temporal features with static metadata. The Temporal Self-Attention Layer applies self-attention to the temporal features following static enrichment. It uses decoder masking to ensure that the temporal dimension is only applied to features preceding it. It also allows TFT to pick up long-range dependencies, which are usually difficult for RNNs to learn. An additional gating layer is also added following the self-attention layer to help with training. The Position-wise Feed-forward Layer uses GRNs to apply additional non-linear processing to the outputs of the self-attention layer. To gain a direct path to the sequence-to-sequence layer skipping over entire transformer block is allowed by utilizing gated residual connections. [31]

Simultaneous prediction of various percentiles (for example, 10th, 50th, and

90th) at each timestep allows TFT to generate prediction intervals on top of point forecasts. Linear transformation at the output of the temporal fusion decoder is used to generate the Quantile forecasts, which can be stated as follows, as stated by Lim et al. [31]:

$$\hat{y}(q, t, \tau) = W_q \tilde{\psi}(t, \tau) + b_q$$

The linear coefficients for the specified quantile q are $W_q \in \mathbb{R}^{1 \times d}$ and $b_q \in \mathbb{R}$. Forecasts are only created for the future, for example, $\tau \in \{1, \ldots, \tau_{\max}\}$.

4 Methodology

4.1 Data

The dataset utilized in this study was collected over the month of May 2022 from the Baltic Sea, capturing AIS transmissions from 3,820 vessels representing more than 10 countries of registry. The distribution of observations per vessel exhibits significant variance. While 3,734 distinct ships recorded at least 100 observations, two highly active vessels recorded in excess of 40,000 observations each. The majority of the collected data originated from passenger vessels registered in Finland.

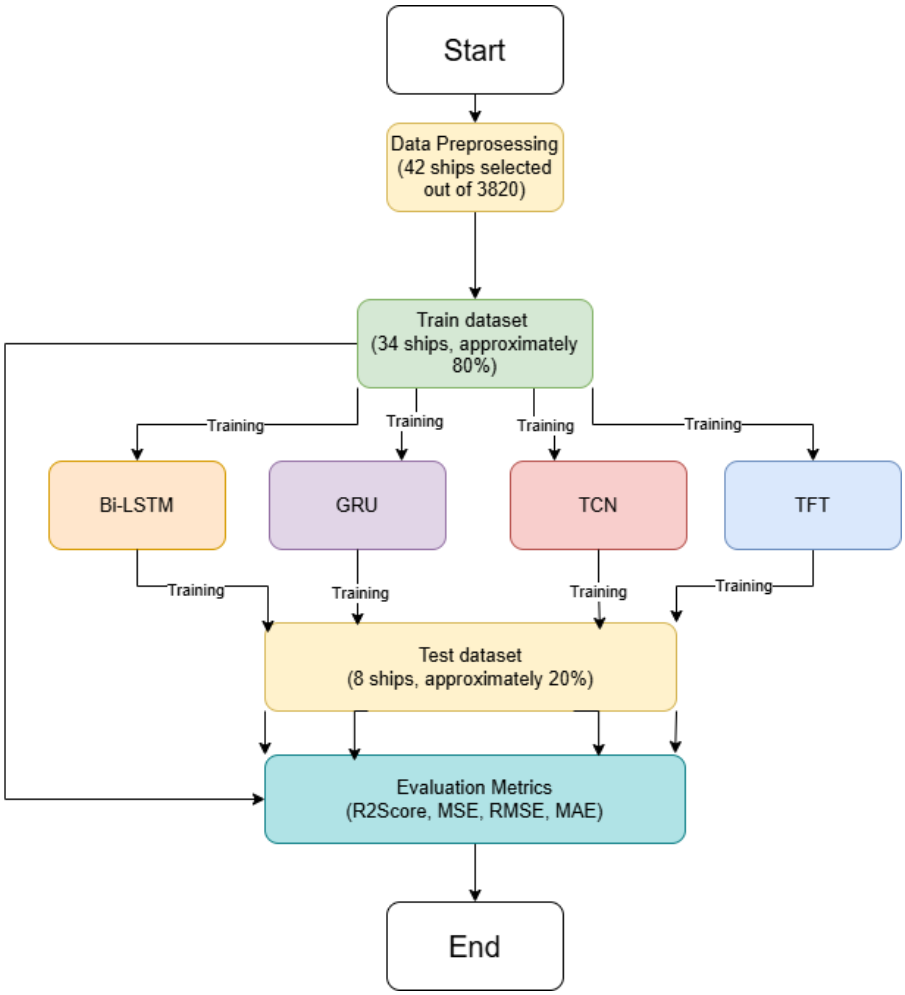
The raw dataset comprises the following features: Timestamp, MSG ID, targetType, MMSI, Latitude, Longitude, Posacc, SOG, COG, shipType, draught, dimPort, dimStarboard, dimStern, Month, Week, IMO, Country, and shipName. A significant portion of these variables are categorical rather than numerical.

To formulate the trajectory prediction task, target variables were constructed by computing the spatial displacements between successive time steps, yielding Latitude Delta (ΔLat) and Longitude Delta (ΔLon). The final model inputs consisted of these spatial deltas, along with Speed Over Ground (SOG), Course Over Ground (COG), and vessel draught. Finally, all input and target features were normalized using a MinMax scaler to scale the values within a consistent range.

4.2 Preprosessing

To ensure a robust training process with sufficient sequence density, only a specific subset of vessels was selected from the dataset. The filtering criteria required each vessel to have recorded more than 30,000 observations and to have remained within the spatial boundaries of 10° to 30° Longitude and 53° to 66° Latitude. Applying these constraints restricted the geographic focus to the Baltic Sea region and resulted in a final cohort of 38 unique ships.

4.3 Methods



The models chosen were Bidirectional Long-Term-Short-Term Memory (Bi-LSTM), Temporal Convolutional Network (TCN), Gated Recurrent Unit (GRU), and Temporal Fusion

Transformer (TFT). All models were hyperparameter-tuned using all of the data from the selected 38 ships.

4.4 Hyperparameter selection and evaluation

The hyperparameters evaluated during the optimization process included the number of hidden units and the dropout rate. The number of units was tested across powers of two: 4, 8, 16, 32, 64, and 128. The dropout rate was evaluated as a binary choice with values of 0.0 and 0.5. Consequently, each model architecture underwent 12 distinct experimental trials.

Each model was constructed with a two-layer architecture. Early Stopping was implemented to monitor the validation loss with a patience of 5, with a maximum limit of 20 epochs per trial. This was done to optimize training and prevent overfitting. Mean Absolute Error (MAE) served as both the training loss function and the primary objective metric for the hyperparameter optimization across all four architectures (Bi-LSTM, TCN, GRU, and TFT).

For the Bi-LSTM, TCN, and GRU models, evaluation metrics—specifically the R^2 score, MAE, and Mean Squared Error (MSE)—were computed at the end of every training epoch. For the TFT model, only the MAE was calculated at each epoch due to its high computational complexity, while the R^2 score and MSE were computed exclusively for the best-performing epoch.

The evaluation metrics utilized to assess model performance are defined as follows:

$$\mathrm{R}^2\mathrm{Score} = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

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

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

R^2Score is used to measure the variance in the data. Scores close to 1 indicate better performance. [26]

MAE is used to measure error in the data by calculating the mean of the difference between predicted and actual values, while not being overly sensitive to outliers. [26]

MSE is also used to measure error in the data, but it penalizes error more heavily due to squaring the difference between predicted and actual values. [26]

RMSE is the square root of MSE, which can make it more intuitive to interpret. [26]

4.5 Model Training

The cohort of 38 unique ships was partitioned into distinct training, validation, and test subsets, which were kept identical across all model evaluations to ensure a fair comparison. Specifically, 26 vessels were allocated to the training set, 6 vessels were reserved as the validation set for hyperparameter optimization, and the remaining 6 vessels were set aside to form the final test set.

For the predictive task, the models utilize a 2-hour window of historical tracking data to forecast the vessel's trajectory over the subsequent 1-hour period. All model training was conducted using a consistent batch size of 256.

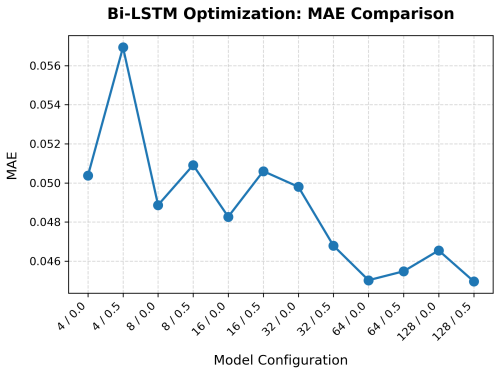
4.6 Hardware and environment

The training was performed using a Windows 11 PC with an AMD Ryzen 5 4600G processor and an NVIDIA GeForce GTX 1660 Super graphics card.

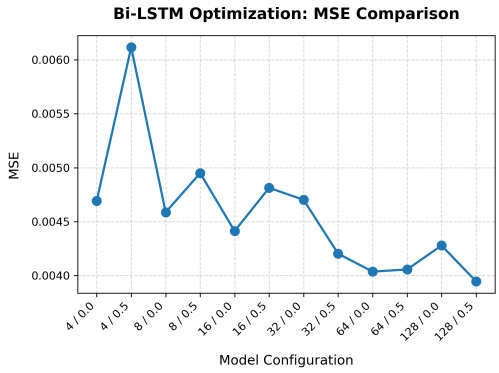
5 Results

5.1 Bidirectional Long-Term-Short-Term Memory

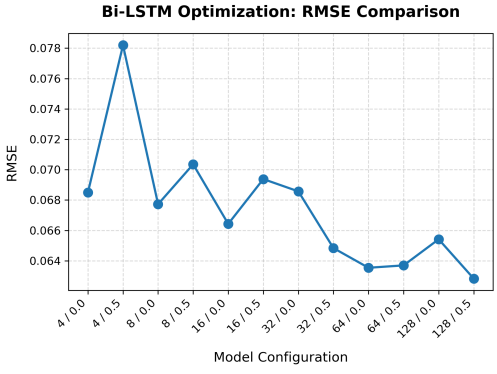
5.1.1 Bi-LSTM hyperparameter optimization results



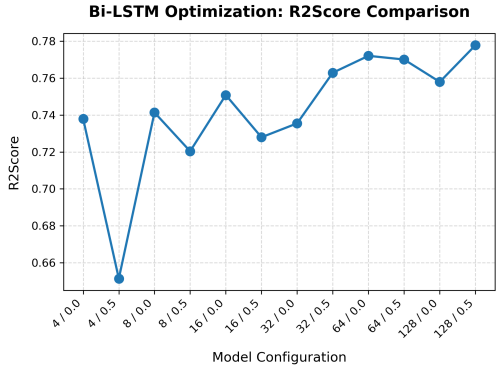
(a) MAE



(b) MSE



(c) RMSE



(d) R2 Score

Figure 5.1: Bi-LSTM validation metrics grid.

The results indicate that both the number of units and the dropout value have

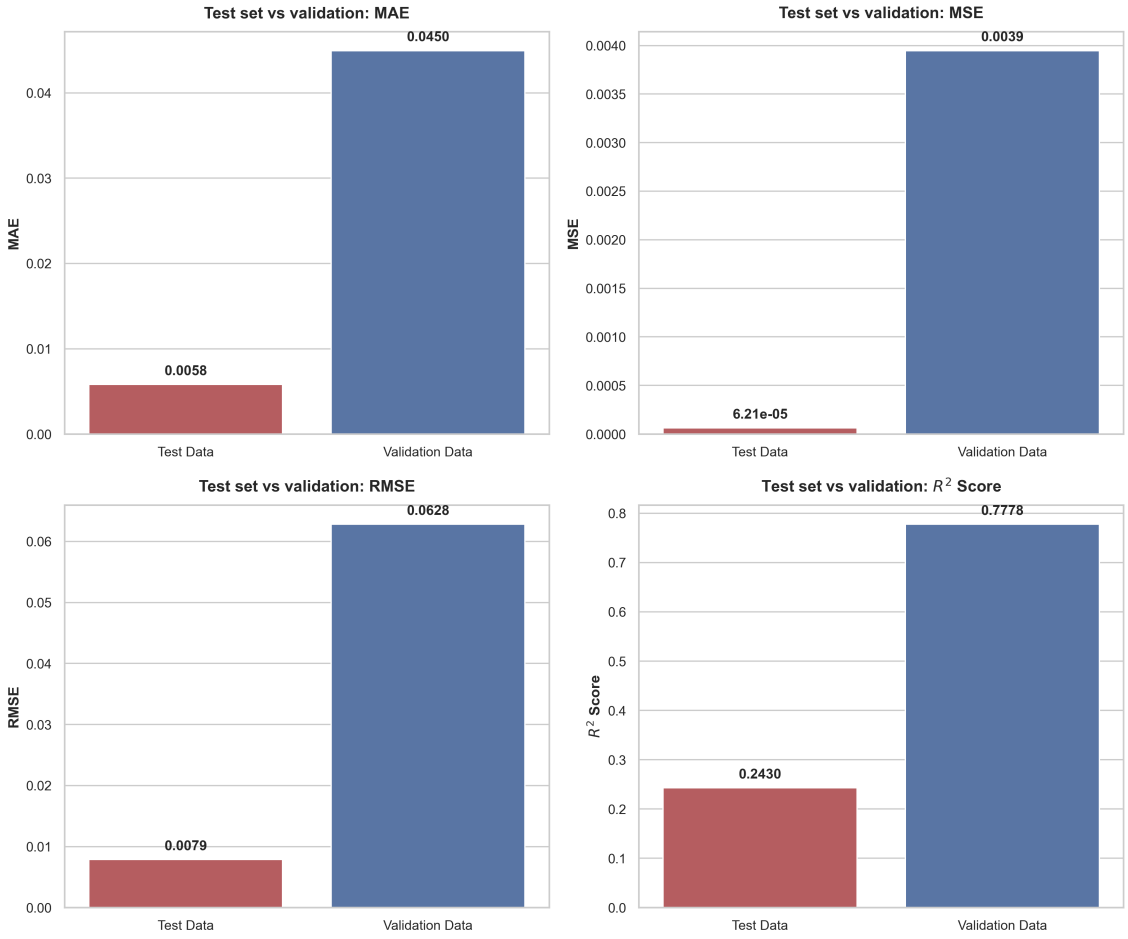
Table 5.1: Bi-LSTM Final results by trial

Trial	Mean Absolute Error	Mean Squared Error	R^2 Score	Root Mean Squared Error	Duration
1	0.0504	0.0047	0.7379	0.0685	8m 50s
2	0.0569	0.0061	0.6514	0.0782	9m 2s
3	0.0489	0.0046	0.7414	0.0677	13m 59s
4	0.0509	0.0049	0.7204	0.0704	13m 8s
5	0.0483	0.0044	0.7508	0.0664	6m 35s
6	0.0506	0.0048	0.7279	0.0694	4m 45s
7	0.0498	0.0047	0.7355	0.0686	13m 48s
8	0.0468	0.0042	0.7629	0.0648	44m 26s
9	0.0450	0.0040	0.7721	0.0635	39m 55s
10	0.0455	0.0041	0.7701	0.0637	40m 13s
11	0.0465	0.0043	0.7579	0.0654	43m 34s
12	0.0450	0.0039	0.7778	0.0628	72m 3s

a clear impact on the model scores. Generally, a higher number of hidden units led to better overall performance. At lower unit capacities (4, 8, and 16 units), the dropout value had a more significant effect on the scores, with the non-regularized models (dropout 0.0) outperforming those with a dropout of 0.5. However, with larger unit amounts such as 32, 64, and 128, this trend inverted, and the models with 0.5 dropout began to outperform the 0.0 dropout baselines. The best overall performance was achieved with a combination of 128 units and a 0.5 dropout rate, yielding an MAE of 0.0450, an MSE of 0.0039, an R^2 score of 0.7778, and an RMSE of 0.0628. The training time for each trial scaled with the model size, ranging from just under 5 minutes to 72 minutes.

5.1.2 Bi-LSTM test results

Figure 5.2: Bi-LSTM test vs validation

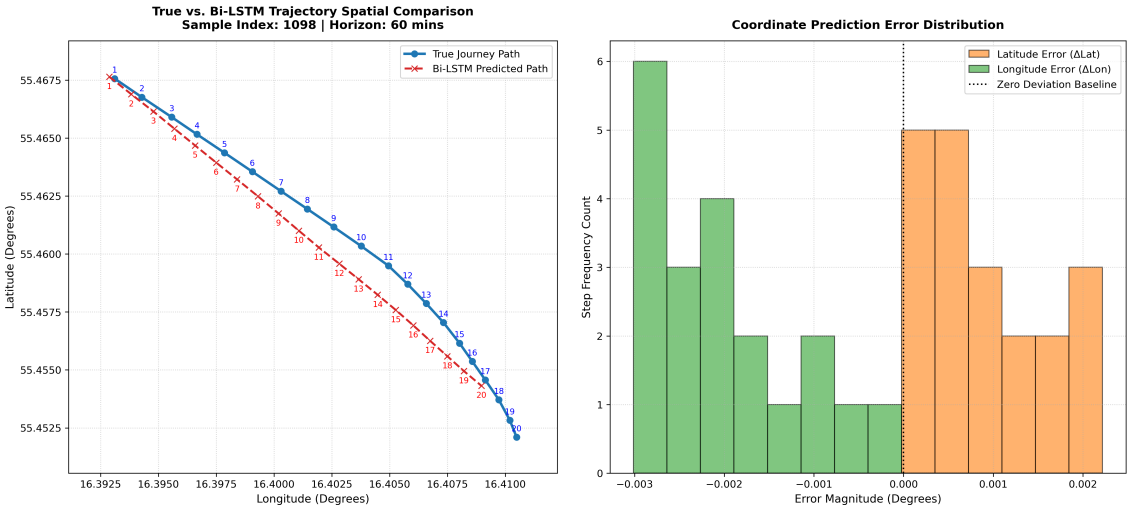


When comparing the test set performance to the optimal validation set scores for the Bi-LSTM model, there are clear improvements in the absolute error metrics: MAE, MSE, and RMSE. Despite these improvements, the R^2 score degrades on the test set compared to the validation set.

This statistical divergence can be explained by the differences in data variance between the two splits. Because the R^2 score is inherently sensitive to the total variance of the ground-truth target distribution, a lower-variance distribution in the test set compresses the baseline denominator. Consequently, even though the model

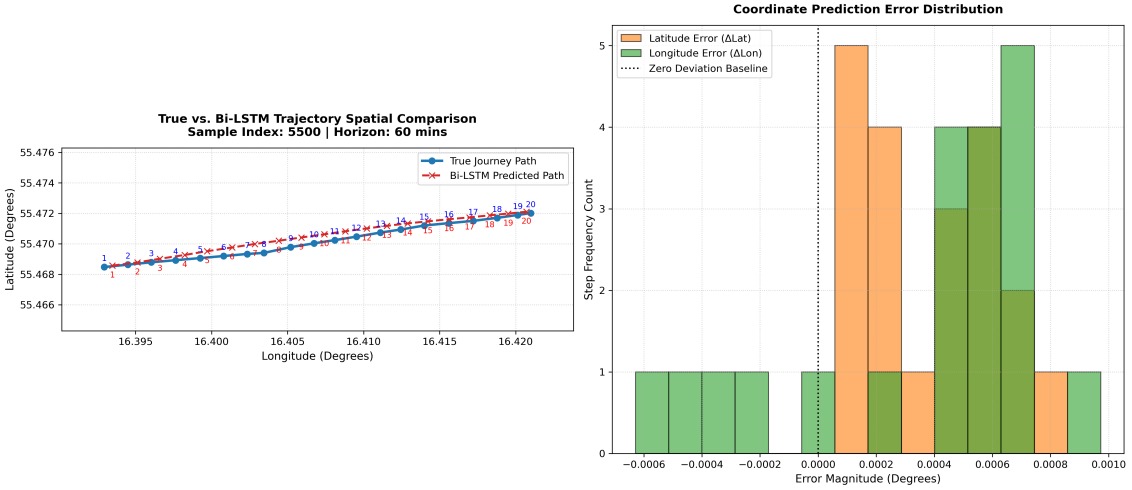
achieves higher absolute accuracy (lower MSE), the remaining errors account for a larger proportion of the smaller total variance in the test data, resulting in a lower relative score.

Figure 5.3: Bi-LSTM Sample I



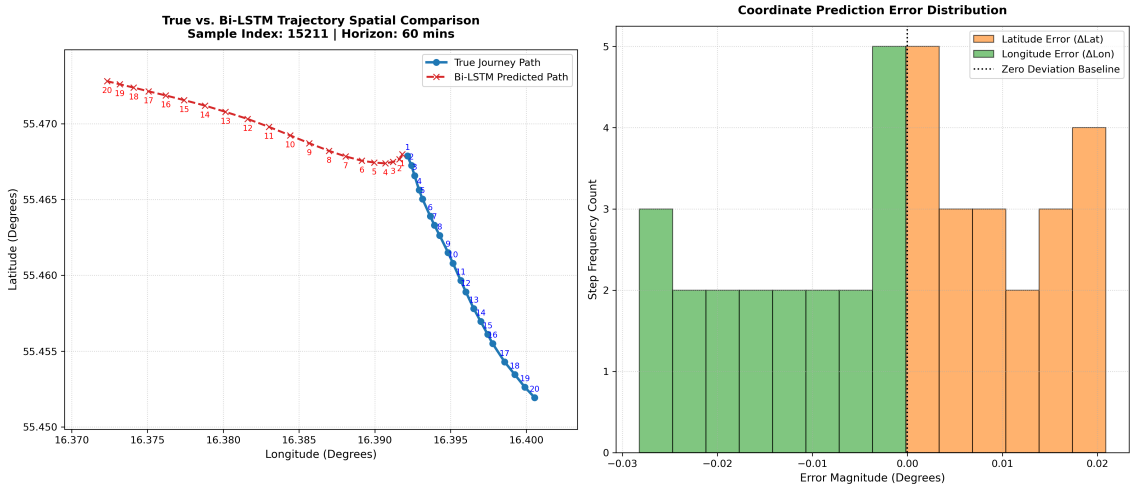
In Figure 5.3, the model successfully predicts the ship movement for approximately half of the forecasting window. After this point, it fails to capture the slight turn executed by the vessel, which leads to an incorrect end-point prediction.

Figure 5.4: Bi-LSTM Sample II



In Figure 5.4, the model is capable of predicting the trajectory relatively well, displaying minimal deviation from the actual path.

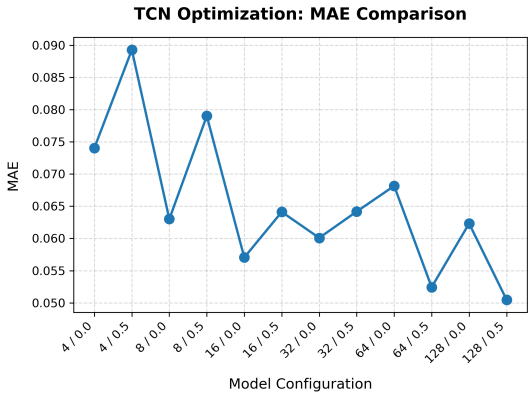
Figure 5.5: Bi-LSTM Sample III

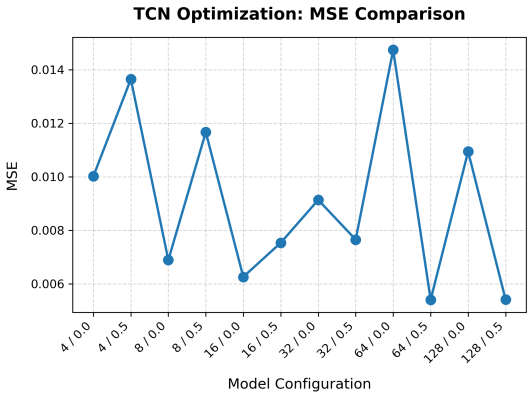


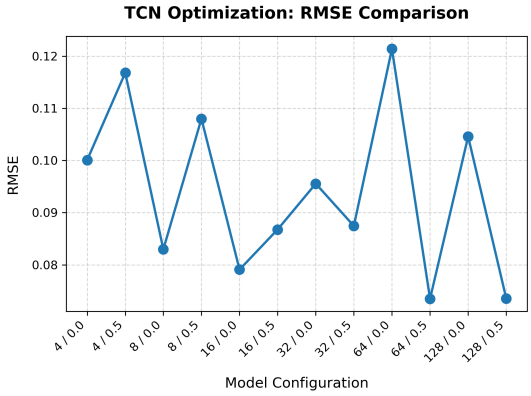
In Figure 5.5, the model correctly identifies that a turn is taking place but fails to predict the correct direction. This directional error results in a complete divergence from the ground-truth trajectory.

5.2 Temporal Convolutional Network

5.2.1 TCN hyperparameter optimization results







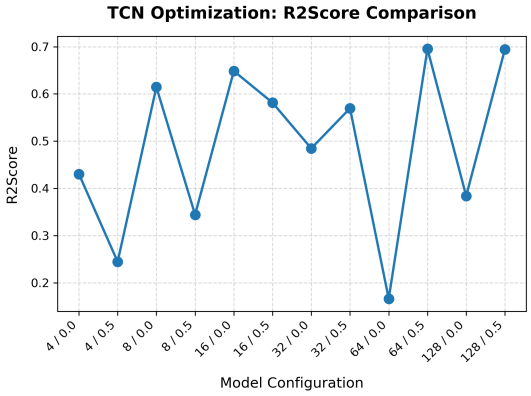


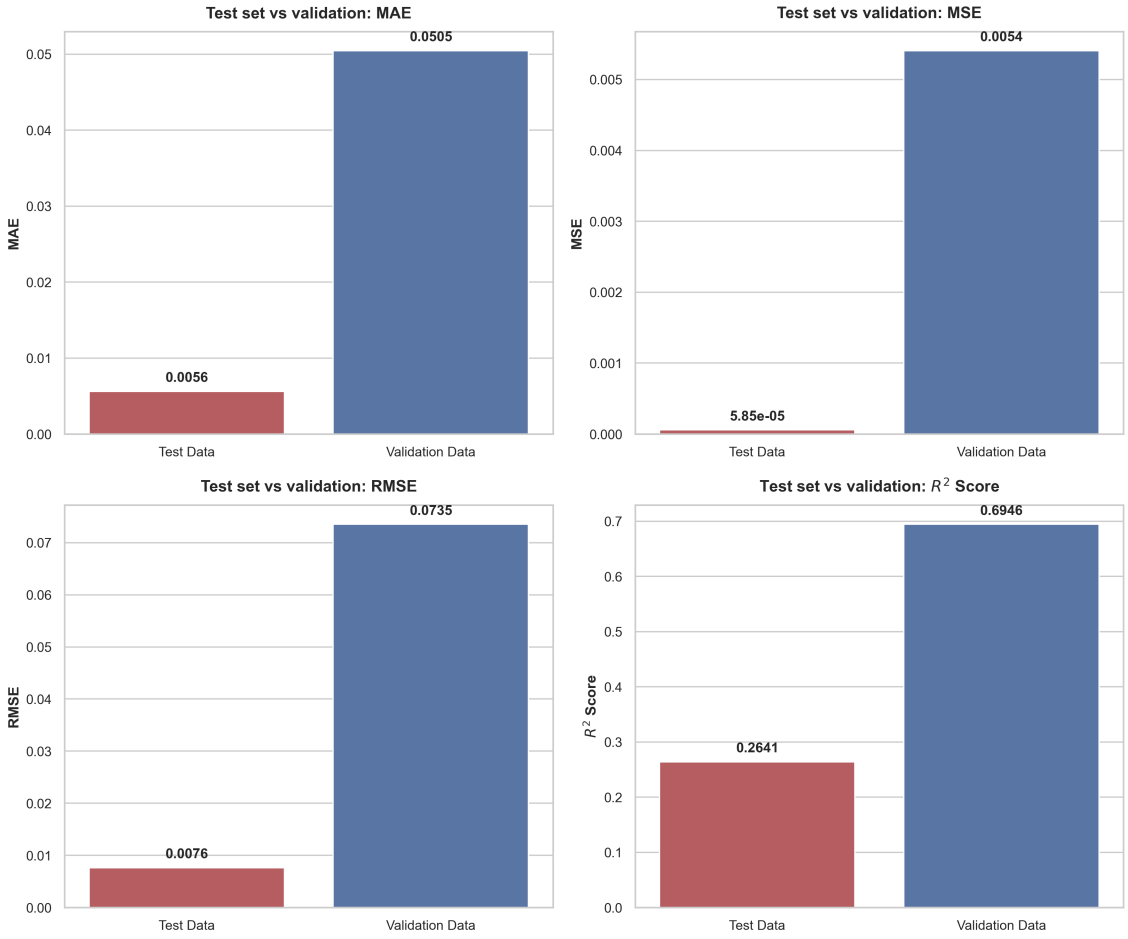
Table 5.2: TCN Final results by trial

Trial	Mean Absolute Error	Mean Squared Error	R^2Score	Root Mean Squared Error	Duration
1	0.0740	0.0100	0.4304	0.1001	10m 7s
2	0.0893	0.0136	0.2444	0.1168	10m 9s
3	0.0630	0.0069	0.6145	0.0830	7m 9s
4	0.0790	0.0117	0.3442	0.1080	7m 8s
5	0.0570	0.0063	0.6485	0.0791	13m 22s
6	0.0641	0.0075	0.5814	0.0868	13m 37s
7	0.0601	0.0091	0.4846	0.0956	20m 28s
8	0.0642	0.0076	0.5696	0.0875	20m 44s
9	0.0681	0.0147	0.1662	0.1214	44m 27s
10	0.0524	0.0054	0.6954	0.0735	44m 52s
11	0.0623	0.0109	0.3838	0.1046	106m 45s
12	0.0505	0.0054	0.6946	0.0735	98m 46s

The results demonstrate the clear effect that the number of units and the dropout rate have on the TCN models. Generally, the models configured with a dropout value of 0.5 outperformed those with a dropout value of 0.0, regardless of the unit size. Increasing the number of units typically led to better performance. The main exception was the model with 64 units and a 0.0 dropout rate, which emerged as one of the worst-performing configurations overall. The top-performing TCN model utilized 128 units and a 0.5 dropout rate, achieving a Mean Absolute Error (MAE) of 0.0505, a Mean Squared Error (MSE) of 0.0054, an R^2 score of 0.6946, and a Root Mean Squared Error (RMSE) of 0.0735. Training duration for each trial ranged from just over 7 minutes to nearly 107 minutes.

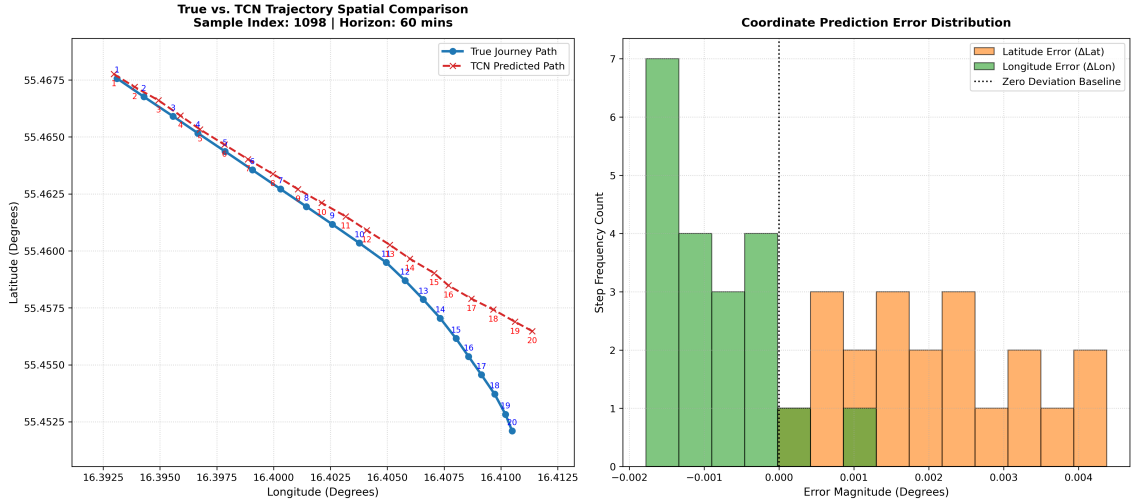
5.2.2 TCN test results

Figure 5.7: TCN test vs validation



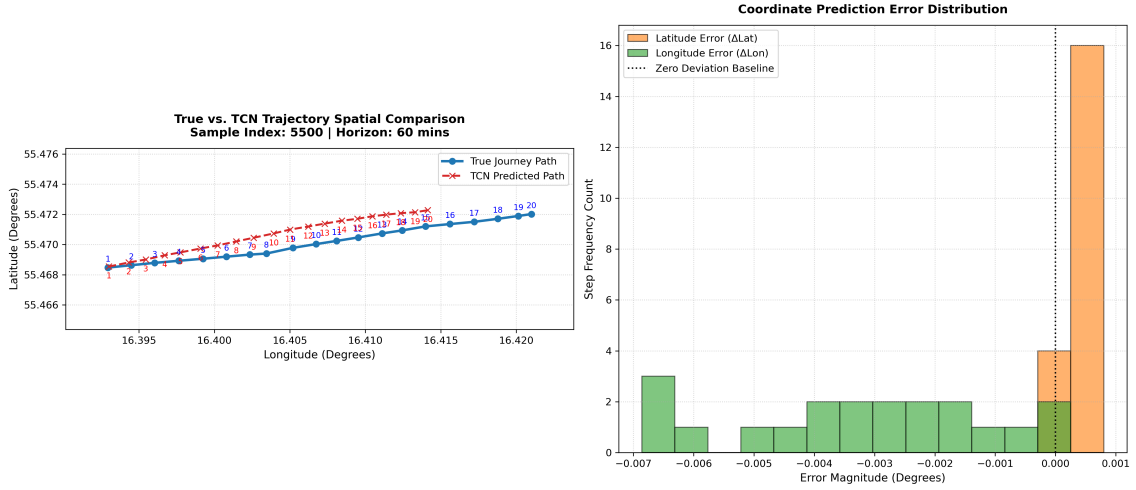
Similar to the Bi-LSTM results discussed in Section 5.2, the TCN model demonstrates an improvement in the absolute error metrics—MAE, MSE, and RMSE—on the test set compared to the validation set. Despite these improvements, the R^2 score similarly degrades when evaluated on the test data.

Figure 5.8: TCN Example A



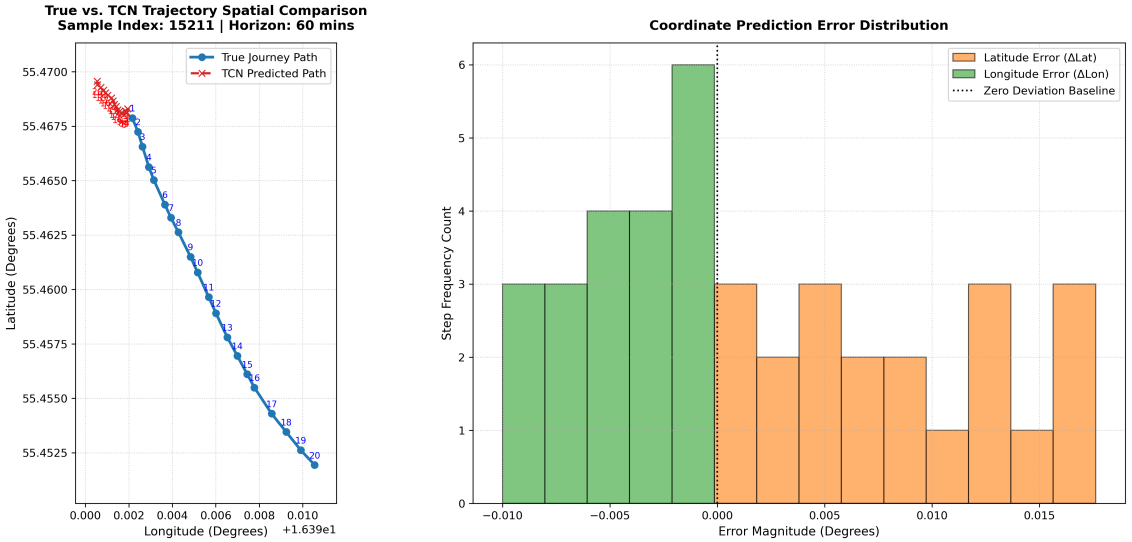
In Figure 5.8, the model is able to predict the general direction of the trajectory. However, it fails to capture a slight turn executed by the vessel, which leads to an inaccurate end-point prediction.

Figure 5.9: TCN Example B



In Figure 5.9, the model correctly identifies the overall direction of the vessel, but the predicted path deviates too far from the actual trajectory, resulting in an incorrect end-point prediction.

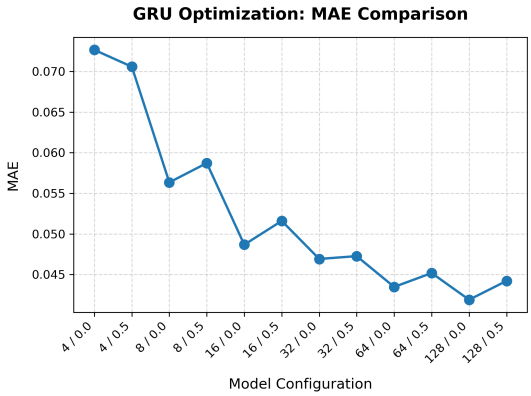
Figure 5.10: Example C

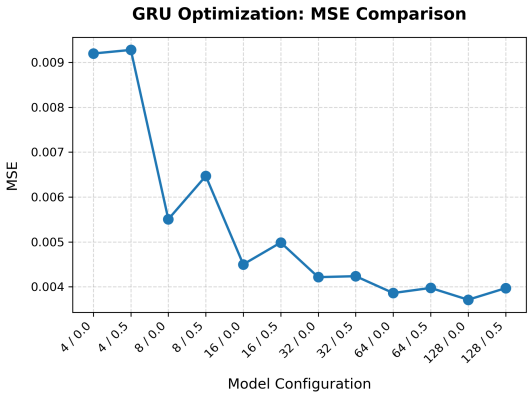


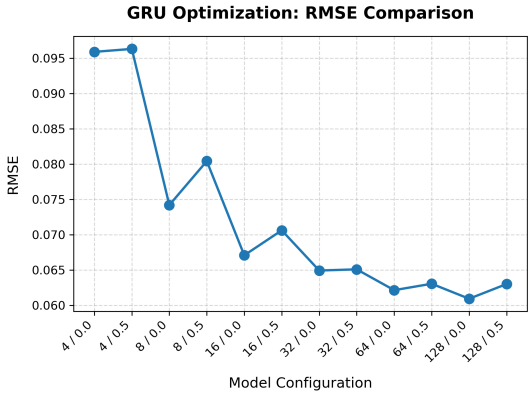
In Figure 5.10, the model completely fails to capture both the turn and the distance traveled by the vessel, resulting in a total divergence from the actual trajectory.

5.3 Gated Recurrent Unit

5.3.1 GRU hyperparameter optimization results







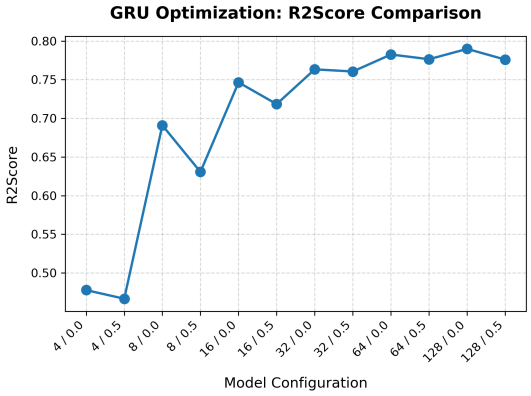


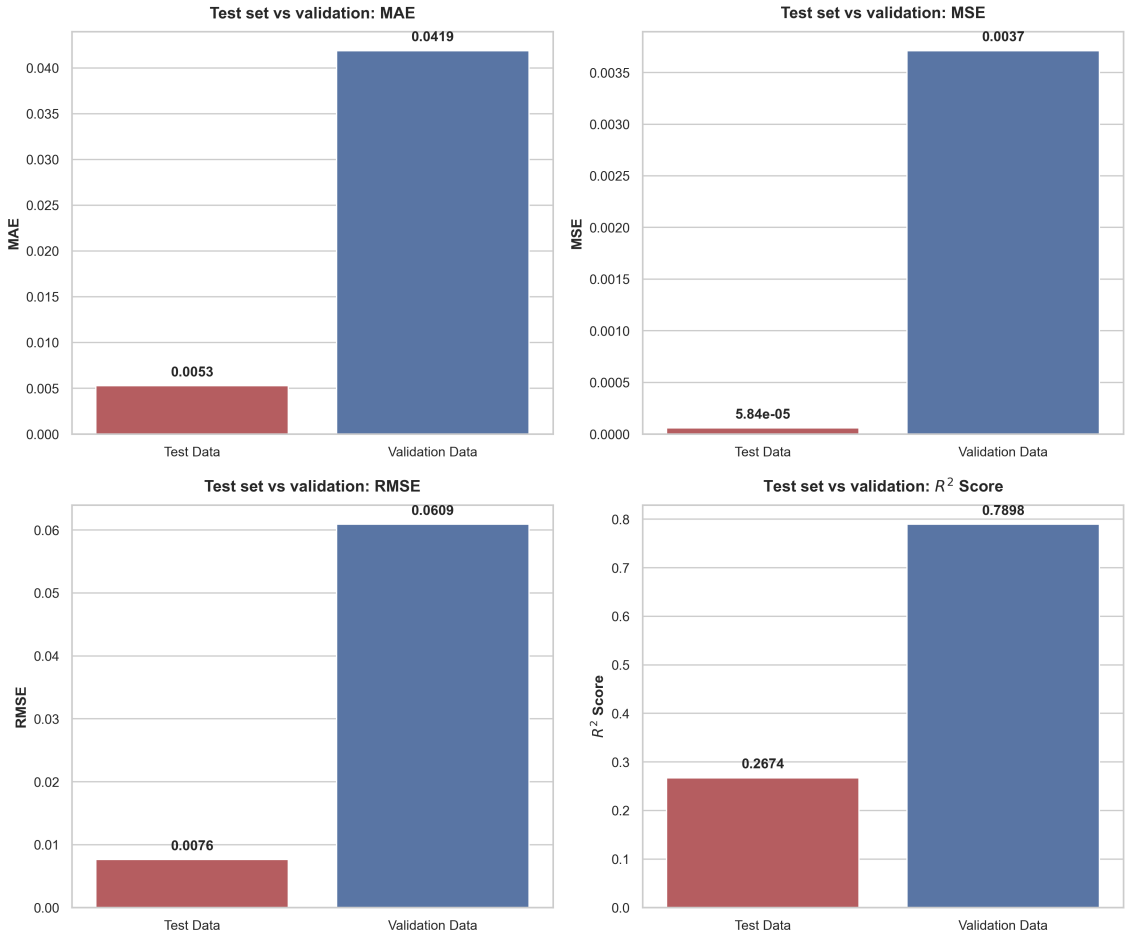
Table 5.3: GRU Final results by trial

Trial	Mean Absolute Error	Mean Squared Error	R^2Score	Root Mean Squared Error	Duration
1	0.0726	0.0092	0.4778	0.0959	4m 19s
2	0.0706	0.0093	0.4666	0.0963	4m 21s
3	0.0563	0.0055	0.6909	0.0742	4m 25s
4	0.0587	0.0065	0.6307	0.0804	4m 23s
5	0.0487	0.0045	0.7465	0.0671	4m 50s
6	0.0516	0.0050	0.7184	0.0706	5m 10s
7	0.0469	0.0042	0.7633	0.0649	14m 22s
8	0.0473	0.0042	0.7606	0.0651	15m 8s
9	0.0435	0.0039	0.7825	0.0621	29m 40s
10	0.0452	0.0040	0.7765	0.0631	26m 21s
11	0.0419	0.0037	0.7898	0.0609	26m 26s
12	0.0442	0.0040	0.7760	0.0630	71m 57s

The choice of hidden units and dropout values clearly affects the performance of the GRU models. A significant improvement in accuracy is observed simply by increasing the unit size from 4 to 8, regardless of the dropout rate. Throughout the evaluations, the models configured with a dropout value of 0.0 consistently outperformed those with a dropout value of 0.5. The best overall performance was achieved using a configuration of 128 units and a 0.0 dropout rate, yielding an MAE of 0.0419, an MSE of 0.0037, an R^2 score of 0.7898, and an RMSE of 0.0609. The training duration for each trial ranged from just over 4 minutes to approximately 72 minutes.

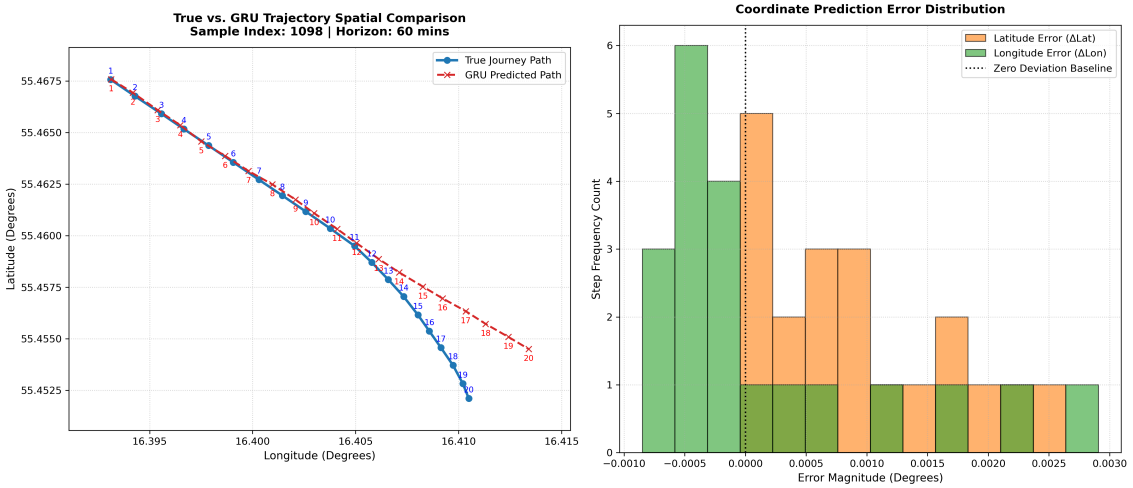
5.3.2 GRU test results

Figure 5.12: GRU test vs validation



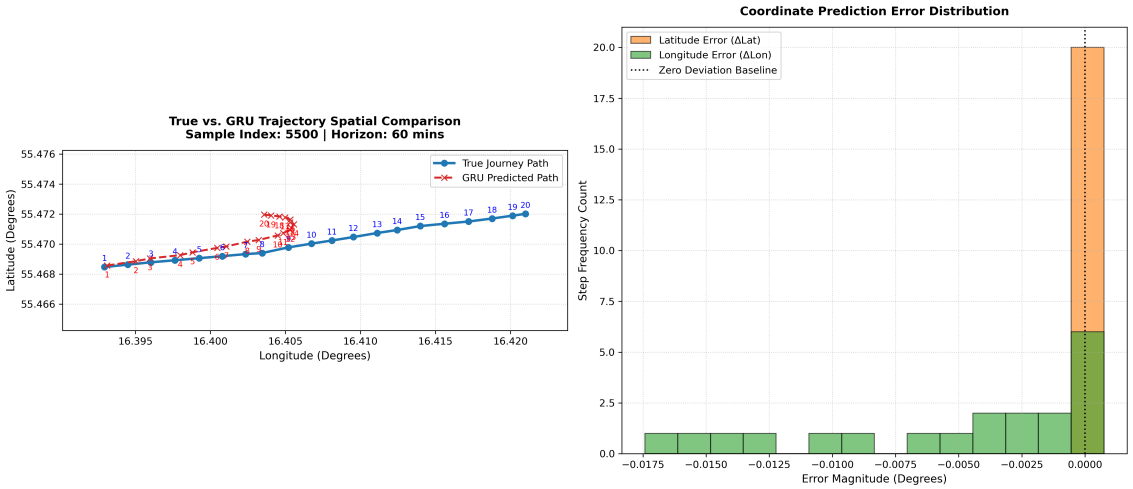
Similar to the trends observed for the Bi-LSTM models in Section 5.2 and the TCN models in Section 5.7, the absolute error metrics—MAE, MSE, and RMSE—all improve on the test set compared to the validation set. However, the R^2 score once again degrades when evaluated on the test data.

Figure 5.13: GRU Example A



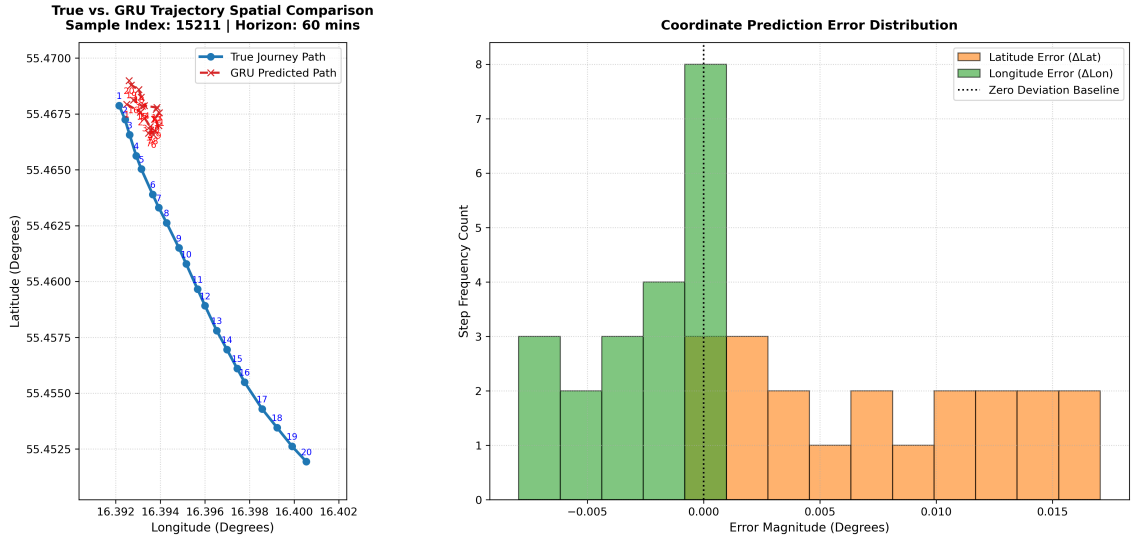
In Figure 5.13, the model is able to track the overall trajectory accurately until approximately the halfway point, where it fails to correctly predict the direction of a turn.

Figure 5.14: GRU Example B



In 5.14 the model fails by predicting a tight turn around halfway point resulting in a total failure.

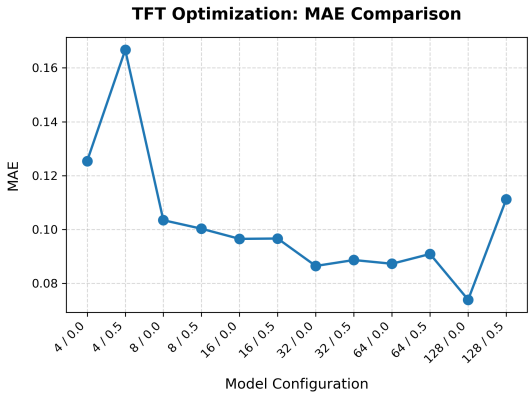
Figure 5.15: GRU Example C

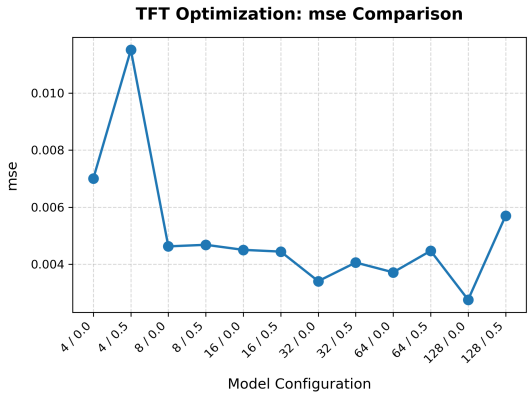


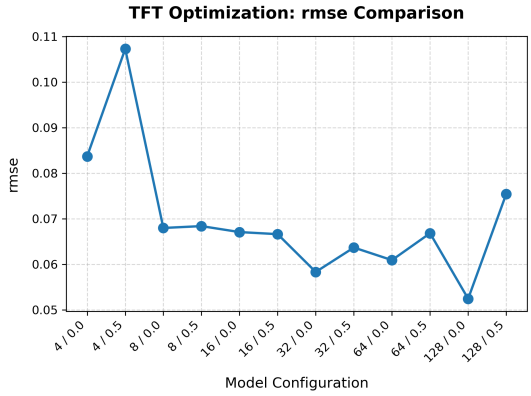
In Figure 5.15, the model fails to predict both the correct direction and the path of the vessel, resulting in a total divergence from the actual trajectory.

5.4 Temporal Fusion Transformer

5.4.1 TFT hyperparameter optimization results







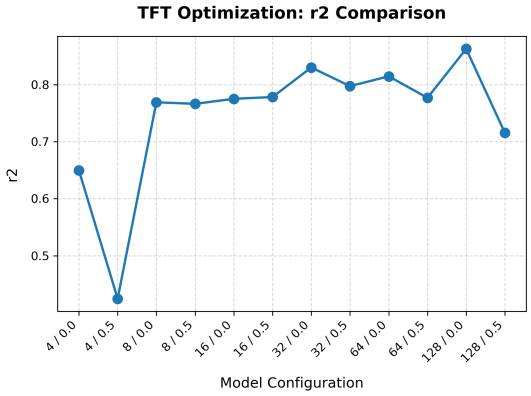


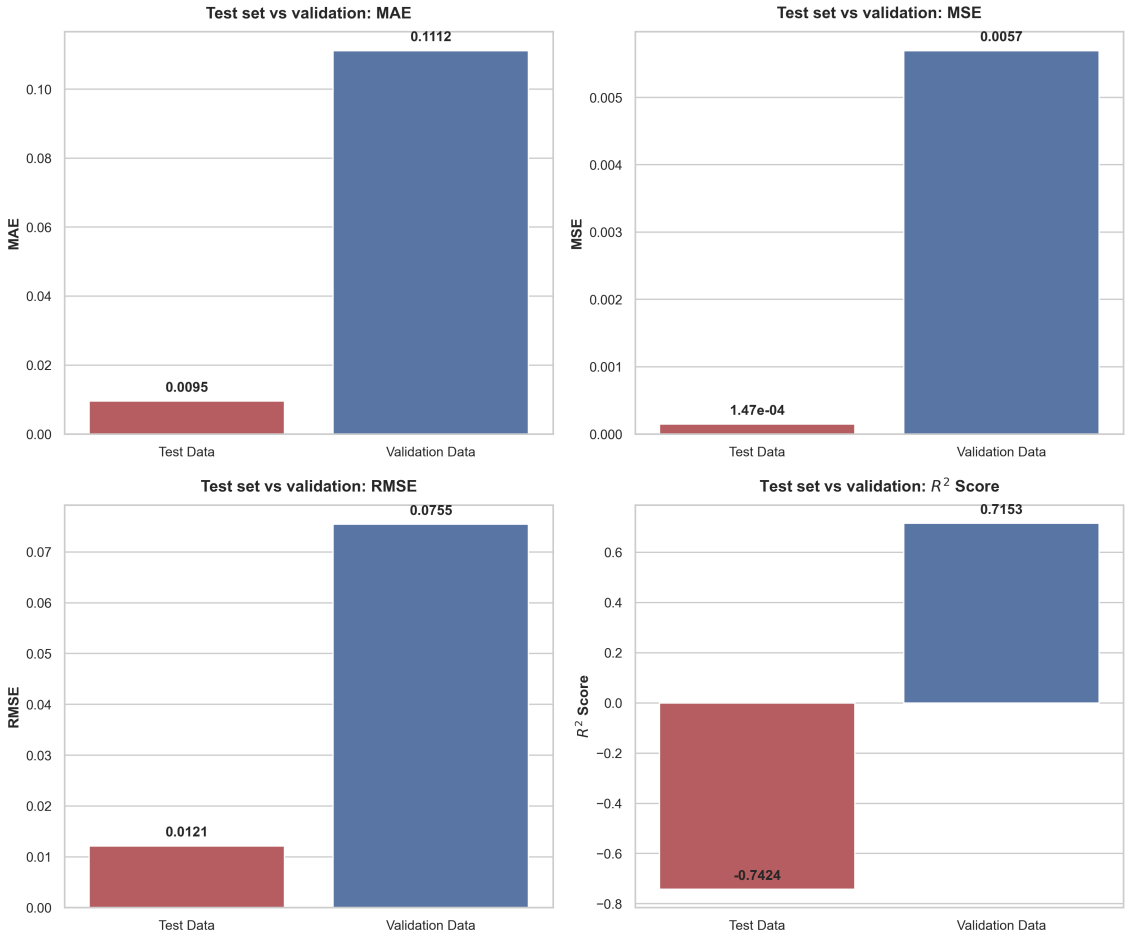
Table 5.4: TFT Final results by trial

Trial	Mean Absolute Error	Mean Squared Error	R^2Score	Root Mean Squared Error	Duration
1	0.1254	0.0070	0.6498	0.0837	186m 13s
2	0.1667	0.0115	0.4246	0.1073	320m 26s
3	0.1034	0.0046	0.7690	0.0680	332m 14s
4	0.1003	0.0047	0.7662	0.0684	334m 21s
5	0.0965	0.0045	0.7750	0.0671	336m 11s
6	0.0966	0.0044	0.7781	0.0666	321m 46s
7	0.0864	0.0034	0.8299	0.0583	335m 44s
8	0.0886	0.0041	0.7972	0.0637	337m 59s
9	0.0872	0.0037	0.8144	0.0609	338m 39s
10	0.0909	0.0045	0.7767	0.0668	238m 40s
11	0.0739	0.0028	0.8625	0.0525	340m 26s
12	0.1112	0.0057	0.7153	0.0755	170m 35s

The choice of hidden units and dropout values has a clear impact on the TFT model performance as well. The models configured with a dropout value of 0.0 consistently outperform those with a dropout value of 0.5. The best hyperparameter combination was achieved using 128 units and a 0.0 dropout rate, yielding an MAE of 0.0739, an MSE of 0.0028, an R^2 score of 0.8625, and an RMSE of 0.0525. The training duration for each trial ranged from just over 170 minutes to approximately 340 minutes, making the TFT by far the slowest model to train among the evaluated architectures.

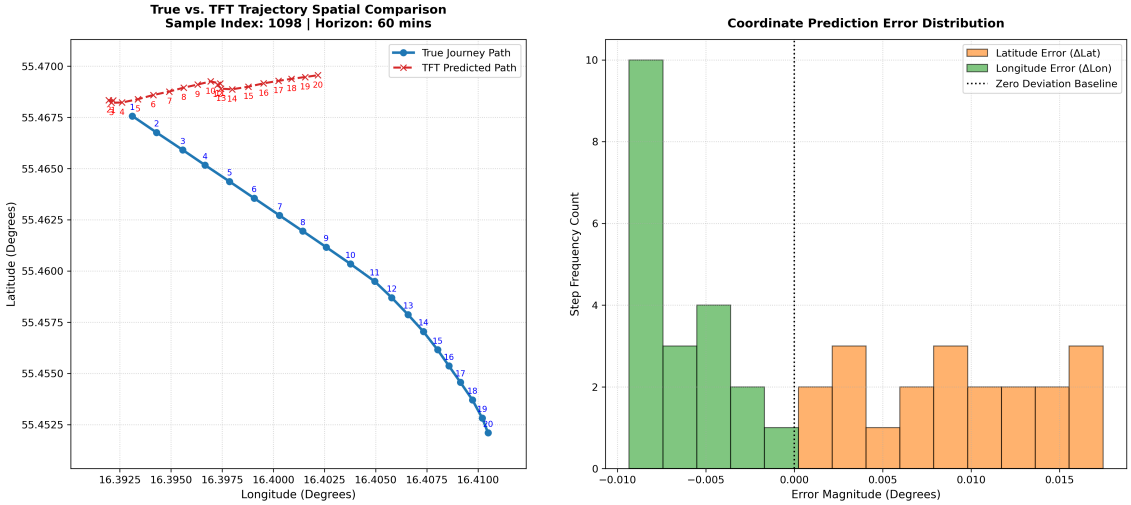
5.4.2 TFT test results

Figure 5.17: TFT test vs validation



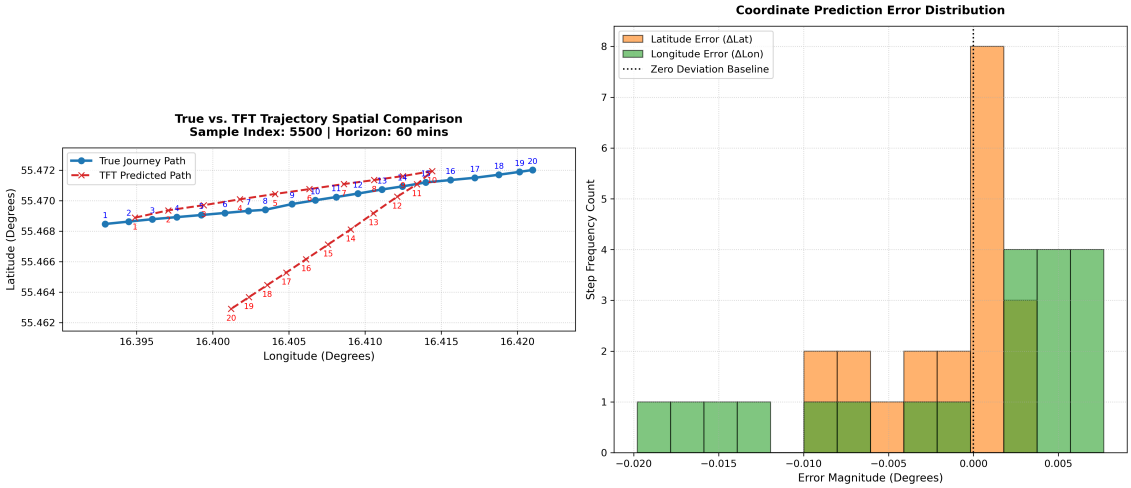
The MAE, MSE, and RMSE scores for the test set all show an improvement over the scores achieved on the validation set. However, the R^2 score is notably worse on the test set compared to the validation baseline.

Figure 5.18: TFT Sample I



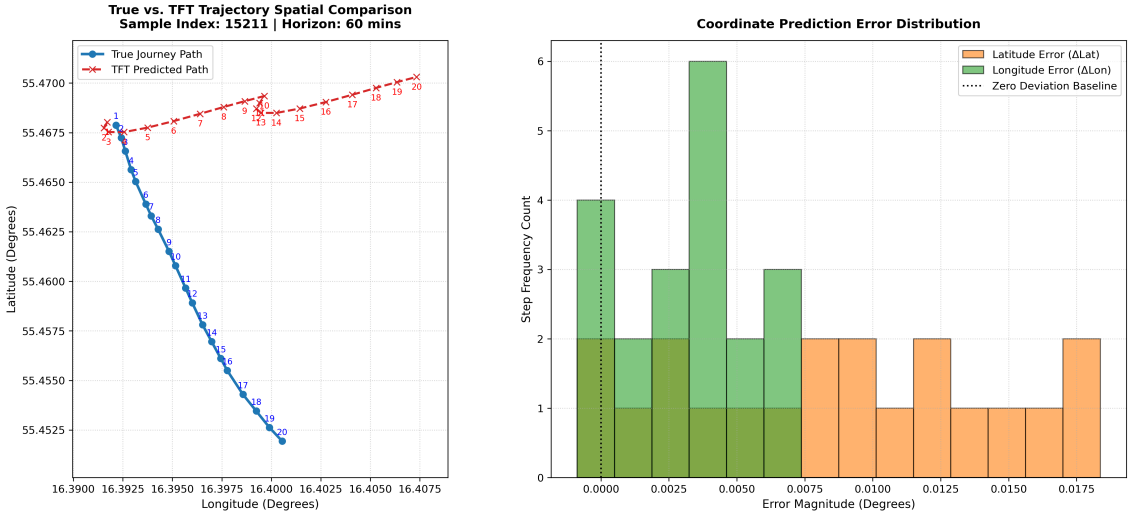
In Figure 5.18, the model fails to predict the correct direction of the trajectory, resulting in a total divergence from the actual path.

Figure 5.19: TFT Sample II



In Figure 5.19, the model accurately tracks the trajectory direction until approximately the halfway point, where it incorrectly predicts a non-existent sharp turn, resulting in a total divergence from the actual path.

Figure 5.20: TFT Sample III



In Figure 5.20, the model is completely unable to predict both the correct direction and the overall trajectory, resulting in a total divergence from the actual path.

5.5 Final Results

For hyperparameter selection, the optimal configuration for both the Bi-LSTM and TCN models was a unit size of 128 combined with a dropout value of 0.5. Conversely, for the GRU and TFT architectures, the best performance was achieved with 128 units and a dropout value of 0.0.

On the validation set, the model performance metrics varied across architectures: the Bi-LSTM model achieved the lowest MAE, while the TFT model excelled across the remaining metrics, delivering the lowest MSE, the lowest RMSE, and the highest R^2 score.

Figure 5.21: Test set model comparsions

Table 5.5: Test set scores

Model	Mean Absolute Error	Mean Squared Error	R^2Score	Root Mean Squared Error
Bi-LSTM	0.0058	0.0001	0.2430	0.0079
TCN	0.0056	0.0001	0.2641	0.0076
GRU	0.0053	0.0001	0.2674	0.0076
TFT	0.0095	0.0001	−0.7424	0.0121

For the test set, the GRU model achieved the lowest MAE at 0.0053. Overall, the GRU, TCN, and Bi-LSTM models demonstrate comparable performance across all metrics, with closely matched MAE, MSE, R^2 score, and RMSE values. The clear outlier is the TFT, which exhibits significantly worse scores than the other

three architectures.

Across all evaluated models, the absolute error metrics—MAE, MSE, and RMSE—unexpecte improved on the test set relative to the validation set, whereas the R^2 scores consistently degraded. Despite these metric alignments, every architecture struggled to reliably predict accurate physical trajectories. Common failure modes included failing to detect subtle turns, overcommitting to extreme changes in heading, overestimating the final trajectory endpoint distance, or misjudging the overall direction entirely.

6 Conclusion

6.1 Results

The hyperparameter optimization results demonstrated that a configuration of 128 units yielded the best performance across all architectures, whereas the optimal dropout value was architecture-dependent. The Bi-LSTM and TCN models achieved their best results with a dropout rate of 0.5, while the GRU and TFT models performed optimally with a dropout rate of 0.0. On the validation set, the Bi-LSTM model achieved the lowest MAE at 0.0044. Meanwhile, the TFT model delivered the best results across the remaining metrics, achieving the lowest MSE of 0.0027, the lowest RMSE of 0.0524, and the highest R^2 of 0.8624.

For evaluation on the test set, the Bi-LSTM and TCN models were configured with 128 units and a 0.5 dropout rate, while the GRU and TFT models were structured using 128 units and a 0.0 dropout rate. Although all architectures demonstrated an improvement in MAE, MSE, and RMSE on the test set compared to the validation baseline, the R^2 scores consistently degraded across all models. Since R^2 score measures sum of the squared errors, it penalizes error much harder than MAE, which is a reason why R^2 score dropped despite MAE improving upon test data. Furthermore, qualitative examination of individual sample trajectories reveals that every model occasionally failed to accurately predict both the final trajectory length and the true heading of the vessel.

These results indicate that the models suffered from overfitting to the validation set during hyperparameter optimization. The models failed to learn the general patterns of ship movement from the training dataset when focusing on optimizing the validation score. This led to the hyperparameter combination that was able to account for the specific noise found in the validation data, but failed when presented with completely unseen test data, as seen with the significant drop in R^2 scores for all models, as seen in the figures comparing actual and predicted trajectories presented in chapter 5.

6.2 Future work

To improve upon these results, several key areas should be prioritized in future research. First, the composition of the training data can be significantly refined. In this thesis, data selection prioritized vessels with the highest number of observations across the entire dataset. Consequently, the chosen ships and trajectories represent highly frequent transits, which inherently biased the dataset toward simple, straight-line routes devoid of major turns or complex maneuvers. Incorporating a more diverse set of trajectories, specifically targeting localized maneuvering zones or varying traffic scenarios, could enhance model robustness.

Second, the structural constraints of the temporal window warrant further experimentation. Utilizing a history length of 2 hours to predict a 1-hour future window inherently restricts the complexity of the observed behaviors; within such short intervals, vessels rarely execute intricate or multi-phase trajectories. Adjusting these ratios or extending the horizons could allow the models to learn more dynamic long-term patterns.

Finally, alternative loss functions should be explored. Because the models were trained using Mean Absolute Error (MAE), the optimization process intrinsically targets the median of the conditional distribution. This mathematical character-

istic often forces the models to produce overly conservative, regularized estimations—such as smooth, straight-line paths—effectively penalizing the model for attempting to predict sharp, low-probability maneuvers and leading it to ignore turns entirely.

References

[1] C.-H. Yang, G.-C. Lin, C.-H. Wu, Y.-H. Liu, Y.-C. Wang, and K.-C. Chen, "Deep learning for vessel trajectory prediction using clustered ais data", *Mathematics*, vol. 10, no. 16, p. 2936, 2022.

[2] H. Li, H. Jiao, and Z. Yang, "Ais data-driven ship trajectory prediction modelling and analysis based on machine learning and deep learning methods", *Transportation Research Part E: Logistics and Transportation Review*, vol. 175, p. 103 152, 2023.

[3] X. Zhang, X. Fu, Z. Xiao, H. Xu, and Z. Qin, "Vessel trajectory prediction in maritime transportation: Current approaches and beyond", *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 19 980–19 998, 2022.

[4] H. Li, H. Jiao, and Z. Yang, "Ship trajectory prediction based on machine learning and deep learning: A systematic review and methods analysis", *Engineering Applications of Artificial Intelligence*, vol. 126, p. 107 062, 2023.

[5] C. N. Burger, T. L. Grobler, and W. Kleynhans, "Discrete kalman filter and linear regression comparison for vessel coordinate prediction", in *2020 21st IEEE International Conference on Mobile Data Management (MDM)*, IEEE, 2020, pp. 269–274.

[6] L. P. Perera, P. Oliveira, and C. G. Soares, "Maritime traffic monitoring based on vessel detection, tracking, state estimation, and trajectory predic-

tion", *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 3, pp. 1188–1200, 2012.

[7] R. Gao, R. Li, M. Hu, P. N. Suganthan, and K. F. Yuen, "Dynamic ensemble deep echo state network for significant wave height forecasting", *Applied Energy*, vol. 329, p. 120 261, 2023.

[8] K.-I. Kim and K. M. Lee, "Deep learning-based caution area traffic prediction with automatic identification system sensor data", *Sensors*, vol. 18, no. 9, p. 3172, 2018.

[9] S. Wang and Z. He, "A prediction model of vessel trajectory based on generative adversarial network", *The Journal of Navigation*, vol. 74, no. 5, pp. 1161–1171, 2021.

[10] X. Chen, J. Ling, Y. Yang, H. Zheng, P. Xiong, O. Postolache, and Y. Xiong, "Ship trajectory reconstruction from ais sensory data via data quality control and prediction", *Mathematical Problems in Engineering*, vol. 2020, no. 1, p. 7 191 296, 2020.

[11] S. Capobianco, L. M. Millefiori, N. Forti, P. Braca, and P. Willett, "Deep learning methods for vessel trajectory prediction based on recurrent neural networks", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 57, no. 6, pp. 4329–4346, 2021.

[12] C. Wang, H. Ren, and H. Li, "Vessel trajectory prediction based on ais data and bidirectional gru", in *2020 International conference on computer vision, image and deep learning (CVIDL)*, IEEE, 2020, pp. 260–264.

[13] D.-w. Gao, Y.-s. Zhu, J.-f. Zhang, Y.-k. He, K. Yan, and B.-r. Yan, "A novel mp-lstm method for ship trajectory prediction based on ais data", *Ocean Engineering*, vol. 228, p. 108 956, 2021.

[14] D. Alizadeh, A. A. Alesheikh, and M. Sharif, "Vessel trajectory prediction using historical automatic identification system data", *the Journal of Navigation*, vol. 74, no. 1, pp. 156–174, 2021.

[15] R. Scheepens, H. van de Wetering, and J. J. van Wijk, "Contour based visualization of vessel movement predictions", *International Journal of Geographical Information Science*, vol. 28, no. 5, pp. 891–909, 2014.

[16] B. Murray and L. P. Perera, "A dual linear autoencoder approach for vessel trajectory prediction using historical ais data", *Ocean engineering*, vol. 209, p. 107 478, 2020.

[17] Y. Suo, W. Chen, C. Claramunt, and S. Yang, "A ship trajectory prediction framework based on a recurrent neural network", *Sensors*, vol. 20, no. 18, p. 5133, 2020.

[18] Z. Xiao, X. Fu, L. Zhang, W. Zhang, R. W. Liu, Z. Liu, and R. S. M. Goh, "Big data driven vessel trajectory and navigating state prediction with adaptive learning, motion modeling and particle filtering techniques", *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 4, pp. 3696–3709, 2020.

[19] L.-z. Sang, X.-p. Yan, A. Wall, J. Wang, and Z. Mao, "Cpa calculation method based on ais position prediction", *The journal of navigation*, vol. 69, no. 6, pp. 1409–1426, 2016.

[20] F. Mazzarella, V. F. Arguedas, and M. Vespe, "Knowledge-based vessel position prediction using historical ais data", in *2015 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, IEEE, 2015, pp. 1–6.

[21] W. Li, C. Zhang, J. Ma, and C. Jia, "Long-term vessel motion predication by modeling trajectory patterns with ais data", in *2019 5th International*

Conference on Transportation Information and Safety (ICTIS), IEEE, 2019, pp. 1389–1394.

[22] J. Venskus, P. Treigys, and J. Markevičiūtė, "Unsupervised marine vessel trajectory prediction using lstm network and wild bootstrapping techniques", *Nonlinear analysis: modelling and control.*, vol. 26, no. 4, pp. 718–737, 2021.

[23] B. Murray and L. P. Perera, "An ais-based deep learning framework for regional ship behavior prediction", *Reliability Engineering & System Safety*, vol. 215, p. 107 819, 2021.

[24] P. Virjonen, P. Nevalainen, T. Pahikkala, and J. Heikkonen, "Ship movement prediction using k-nn method", in *2018 Baltic Geodetic Congress (BGC Geomatics)*, IEEE, 2018, pp. 304–309.

[25] S. Thombre, Z. Zhao, H. Ramm-Schmidt, J. M. Vallet García, T. Malkamäki, S. Nikolskiy, T. Hammarberg, H. Nuortie, M. Z. H. Bhuiyan, S. Särkkä, and V. V. Lehtola, "Sensors and ai techniques for situational awareness in autonomous ships: A review", *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 1, pp. 64–83, 2022. DOI: 10.1109/TITS.2020.3023957.

[26] F. Farahnakian, "Digital maritime monitoring: Enhancing situational awareness in shipping traffic using ai-based models", Ph.D. dissertation, University of Turku, 2026.

[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural Computation*, vol. 9, pp. 1735–1780, Nov. 1997. DOI: 10.1162/neco.1997.9.8.1735.

[28] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection", in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 156–165.

[29] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches", in *Proceedings of SSST-8, eighth workshop on syntax, semantics and structure in statistical translation*, 2014, pp. 103–111.

[30] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling", in *NIPS 2014 Workshop on Deep Learning*, 2014.

[31] B. Lim, S. Ö. Arık, N. Loeff, and T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting", *International journal of forecasting*, vol. 37, no. 4, pp. 1748–1764, 2021.