Studiengang: Wirtschaftsinformatik

Developing Vessel Behavior Predictions for Maritime Anomaly Detection

Masterarbeit

Fakultät II
Department für Informatik
Abteilung Systemanalyse und -optimierung

Themensteller:	Prof. Dr. Axel Hahn
Betreuer:	Matthias Steidel
2. Betreuer:	Dr. Arto Niemi

vorgelegt von:

Finn-Matthis Minßen
Schenumer Weg 9
6041507
26441 Jever
015141460007
E-Mail: finn-matthis.minssen@uni-oldenburg.de

Abgabetermin: 18. Mai 2023

Abstract

This thesis proposes an approach to predict vessel tracks in waterways using a Bi-directional Long Short-Term Memory (Bi-LSTM) model as well as a transformer model. For this purpose, a concept is developed that combines positions of buoys along the Elbe and Weser rivers delimiting the waterway with Automatic Information System (AIS) data. Additionally, tide information as well as weather information will be added to the created dataset. These data are then used to train both models to predict vessel tracks. During evaluation, both models will be compared with each other as well as with a linear prediction. Furthermore, the influence of the tide data and weather information to the accuracy of the predictions is evaluated. Building upon the developed concept and the prediction models, this thesis investigates whether the predictions of the model that makes the most accurate predictions can be used to mark tracks that show anomalous vessel behavior. Therefore, predictions are then made based on a new created dataset in the area of interest and deviations that exceed a predefined threshold are flagged as anomalous. Results show that with the developed concept, vessel tracks inside waterways can be accurately predicted.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Problem Description

The shipping industry is a critical component of the global economy, with around 80% of global trade by volume and 70% by value carried out by sea [1]. This high volume of traffic can lead to high density areas, especially in coastal regions, since the routes that can be navigated by vessels are limited. Accidents like the one involving a container vessel that ran aground off the German island Wangerooge can pose major risks to the environment and disrupt supply chains [2]. Other incidents, such as the collision of a cargo vessel with a wind farm in the North Sea, have in the past disrupted energy supplies and thus pose further risks to the energy infrastructure [3]. To avoid these types of incidents, maritime surveillance and Maritime Situational Awareness (MSA) are becoming increasingly important. To be able to develop such awareness, predictions of where a vessel is likely to be in the near future and whether there are any anomalies in the vessel's behavior are very valuable [4]. An anomaly is generally understood as a deviation from a standard or expected value [5]. In the context of this thesis, an anomaly occurs when a deviation from a prediction exceeds a specific threshold. At the basis of detecting anomalous vessel behavior, are the expected track or coarser patterns, which can be predicted using data-driven models based on historical data [6][7]. Current research methods for predicting vessel tracks or anomalous vessel behavior use data provided via the Automatic Information System (AIS) as a foundation. This data includes the position as well as the course and speed of the respective vessel [8]. However, other factors such as weather conditions, tides, regional geographical characteristics as well as the vessel type itself also influence the track followed. This is not yet sufficiently considered in research [7]. As Zhang [8] points out, identifying and fusing these heterogeneous data from multiple sources to predict more accurate vessel tracks remains a research challenge.

Derived from the problem description, the following questions arise:

1. How can vessel tracks be predicted by incorporating multiple data sources?
2. To what extent does adding multiple data sources improve prediction accuracy?
3. How can the developed method be used to flag anomalous tracks?

1.2 Aim of this Thesis

In this thesis, two data-driven models are developed for iterative predictions of vessel tracks in waterways. These models are to be trained with different combinations of features. These features are extracted from AIS data and combined with waterway position, tidal information and weather characteristics. Within this thesis, a concept is presented on how these data can be combined so that data-driven models can be trained. The results of each model are then evaluated against each other as well as against a linear prediction in order to find the model that provides the most accurate predictions. The models will be trained and tested with data from a region along the West German coast as well as the Weser and Elbe rivers.

In order to evaluate the results of this thesis, requirements for the concept as well as for the data-driven models are specified. The requirements for the concept include the combination of historical vessel data, weather information, tide data and sea chart information. These datasets have to be combined to create a comprehensive training dataset for the data-driven models, enabling them to learn patterns and relationships that contribute to accurate predictions of future vessel tracks. The data-driven models will also have to meet certain requirements. They should be capable of predicting vessel tracks specifically in coastal areas. Coastal areas possess unique characteristics, such as changing tides, complex currents and varying wave patterns, which significantly influence vessel movements compared to open seas. Hence, the models developed for coastal areas need to be trained and evaluated using data from coastal areas to capture these distinct dynamics accurately. This data has to be generated by the concept developed in this thesis. The data-driven models should produce predictions that closely align with the actual vessel tracks since achieving high accuracy is essential to ensuring the reliability and usefulness of the predictions.

This thesis further investigates whether the predictions of the model that delivers the most accurate predictions can be used to mark tracks that indicate anomalous vessel behavior. When evaluating this use case, data is selected from a period that was not considered when training the model. Predictions are then made based on this data, and deviations that exceed a predefined threshold are flagged as anomalous.

1.3 Outline

This structure of this thesis is displayed in table 1:

Chapter	Topic	Content Summary
1	Introduction	Introduction to the topic covered and overview of the aim of this thesis. Further research questions are raised that will be answered with this thesis.
2	Maritime Data	Background on Maritime Data. This includes AIS data, weather data and information about waterways.
3	Machine Learning	Background on Machine Learning, specifically on Recurrent Neural Networks, Long Short-Term Memory cells, Attention Mechanism and Transformer.
4	Related Work	Presents the current research in the fields of Vessel Track Prediction and Anomaly Detection in the Maritime Domain.
5	Concept	Describes a concept to combine AIS data, data about positions of waterways, tide data and weather information for predicting vessel tracks. Further a concept to flag anomalous vessel tracks is also described.
6	Evaluation	Evaluates and discusses the developed concepts. This involves generating a dataset based on the concept, which is then used to train data-driven models that make predictions about vessel tracks. These models are then compared with each other. Further, it is tested whether anomalous vessel behavior can be detected using the model that makes the most accurate track predictions.
7	Summary and Outlook	Summaries the results by answering the research questions. Furthermore, an outlook on interesting research possibilities resulting from this thesis is given.

1 Outline Thesis

2 Maritime Data

In the maritime domain, three main categories of data sources exist. The first category comprises data received by sensors in a specific coverage area. Among these data are time stamps, vessel-specific information, vessel characteristics as well as geodata information. These sensor data can be further categorized into active (radar, sonar) and passive (AIS). While radar and sonar data are in general not publicly available, AIS data is receivable with an appropriate receiver. [9]

The second category contains authorized database information about crews or vessel cargo that are sent through SafeSeaNet or the West European Tanker Reporting System. This information is used by authorities to keep track of shipping and ensure compliance with safety and environmental regulations. In general, this data is also not publicly available. [9]

The third category contains data from internet data sources that are publicly available. These data can be provided by ports or from services like Marine Traffic [10]. This also includes weather data or tidal information that is publicly provided by institutions or companies.

In this chapter we take a deeper look into AIS data since it is free available and mainly used in vessel prediction tasks and into maritime weather data and waterways because both also have a great influence in marine traffic.

2.1 Automatic Identification System (AIS)

Particularly important in the maritime domain is data provided by AIS. AIS was developed to avoid collisions at sea and is used to identify and locate vessels in real time. It is based on automatic data exchange between vessels regarding their characteristics and the positions between them and other vessels in the nearby areas. AIS data is collected, on the one hand, from AIS base stations that monitor traffic in specific areas and, on the other hand, from satellites that collect data on a global scale. Because it is mandatory for vessels above 300 GRT (Gross Register Tonnage) to use AIS transmitter, this data can be used to obtain a view of shipping traffic all over the world. Additionally, the data is updated frequently, so individual routes of vessels can be displayed. AIS data can be further divided into dynamic, static and voyage-specific data. Dynamic data contains information on location, course, speed and navigational status and is transmitted at intervals of two to ten seconds. Static data may include the identification number, vessel type, name and call sign of the vessel, while voyage-specific data includes information

about the destination and estimated time of arrival. These data are updated every 6 minutes. [11]

Table 2 describes the transmitted AIS static information in more detail, while table 3 describes dynamic AIS information.

AIS Static Information	
IMO	Vessel identification number
Callsign	International radio call sign - assigned to the vessel by its country of registry
Name	Name of the vessel
Type	Type of vessel/cargo
Dimension	Dimensions of a vessel, to nearest meter
Location of Antenna	Location of positioning system's antenna on board the vessel
Positioning System	Type of positioning system, such as GPS, DGPS
Draught	Draught of ship
Destination	Where a vessel is heading
ETA	Estimated Time of Arrival at destination - UTC month/date hour:minute

2 AIS Static Information [11]

AIS Dynamic Information	
MMSI	Unique nine-digit identification number of a vessel
Navigation status	e.g., “at anchor”, “under way using engine(s)”, “not under command”, etc.
Rate of turn	right or left, in degrees per minute
Speed over ground (SOG)	vessel’s speed in 1/10 knots
Course over ground (COG)	vessel’s course relative to true North
Position	latitude/longitude of a vessel
True Heading	vessel’s course in degrees

3 AIS Dynamic Information [11]

When using AIS data, for example for data analysis, there are several weaknesses that need to be considered. First of all, the AIS information must be transmitted through an AIS transponder that can be switched off by the vessel master [9]. Fishing boats, for example, sometimes turn off the transponder to hide from the competition where they are fishing [12]. Accordingly, vessel tracks can be incomplete and important parts of messages can be missing. Furthermore, data compiled from different sources may contain irregularities, for example regarding time, which means that the route of a vessel cannot be tracked quite correctly [9]. Furthermore, data coverage is not equally good across all areas. Especially in areas with a high vessel density such as the North Sea or the Baltic Sea, the satellites may not be able to cover all transmitted data due to limited coverage [9]. The fact that AIS data works as an open-sourced transmitting system is simultaneously the main advantage and disadvantage of the system, since the AIS data can be spoofed or hijacked. Accordingly, when working with AIS data, it is necessary to validate the data and not just rely on a single source [13].

2.2 Weather Data

Maritime weather data is another valuable source of information for monitoring and preventing incidents in the maritime domain. Weather conditions can significantly impact maritime traffic, with severe weather leading to delays, diversions and even accidents [9]. Weather data is provided through different sources like the European Centre for Medium-Range Weather Forecasts (ECMWF) [14]. The ECMWF is a part of Copernicus, which is the Earth observation component of the European Union's Space program [15]. It is managed by the EU and the European Space Agency (ESA) and provides data based on satellite observations and seaborne measurements systems. For the maritime context, the ERA5 weather dataset, in particular, is of great value [16]. The dataset provides a comprehensive record of the Earth's climate from 1940 to the present day, with data available at latitude-longitude grids with 0,25° x 0,25° resolution for atmosphere features and 0,5° x 0,5° for ocean waves. Era5 includes hourly estimations for a range of atmospheric variables, including temperature, pressure, wind, humidity, precipitation and radiation, which can be used for understanding and modeling the Earth's climate system. The data is collected using a range of sources, including satellites, radiosondes and surface-based observations. It is then assimilated into an atmospheric model, which combines the observational data with the model output to produce a consistent and comprehensive record of the Earth's climate. The data is freely and openly accessible to users and can be downloaded from the Copernicus servers [15].

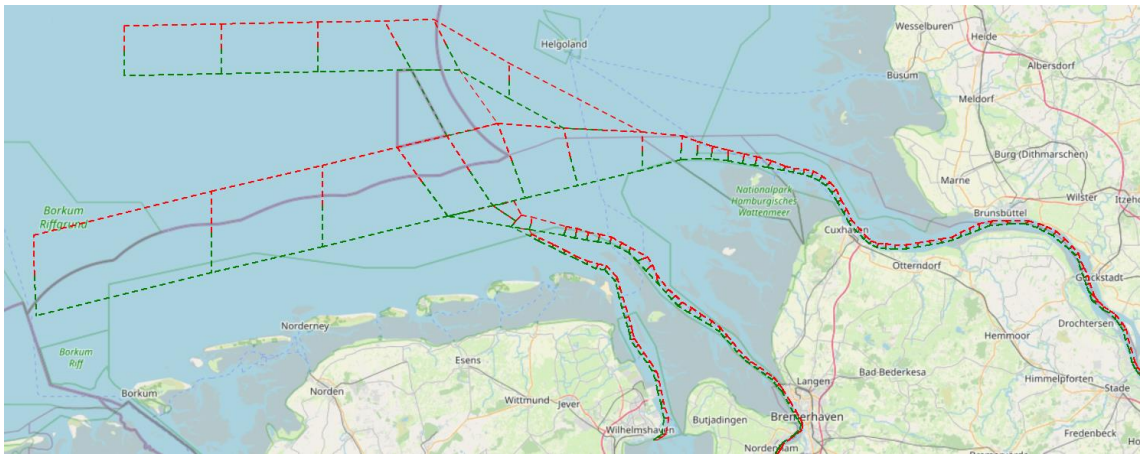
For this thesis, which uses the data introduced above in the maritime context, especially the wind speed, wind direction as well as the wave height are of interest. Wind speed and wind direction data are based on a global atmospheric reanalysis. The wind speed, which is specified in meters per second, is calculated based on the vector wind, which is the horizontal wind component in both the eastward and northward directions. The wind direction is the direction from which the wind is blowing and is provided in degrees. The significant wave height is estimated based on the wave model Waves for the Atlantic and the Mediterranean (WAM). The WAM model calculates the wave spectrum based on wind fields, which are in turn derived from atmospheric data. The wave spectrum describes the distribution of wave energy across different wavelengths and directions. The significant wave height is defined as the average height of the highest one-third of the waves in a given wave spectrum and is measured in meters.

In addition to the weather itself, the tide can also have a major impact on maritime traffic. For example, at certain times waterways may not be navigable due to a low tide, or may only offer sufficient depth for certain vessels. This tide information is provided by the

Copernicus Marine Environment Monitoring Service (CMEMS), by regularly measuring and recording water levels at specific buoys in the sea relative to a fixed reference [17]. The water levels are recorded in 10-minute intervals and can be downloaded freely.

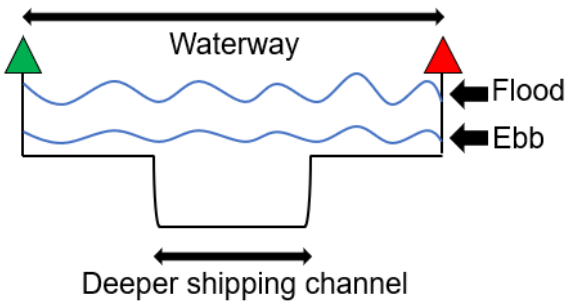
2.3 Waterways

In order for vessels to navigate safely along coastal areas as well as on rivers, waterways delimit the navigable waters. These waterways are usually marked on both sides with buoys as defined by the International Association of Lighthouse Authorities (IALA). A vessel approaching a port has the green buoys on the right side and when leaving the port, the red buoys on the right side. Figure 1 shows a section of the waterways along the German coast, including the entrances to the Weser, Elbe and Wilhelmshaven.



1 Waterways along the Area of Interest

The use of waterways helps to structure traffic and reduce the risk of collisions in high-density areas. The buoys along the waterway also ensure a minimum depth within the waterway, allowing vessels to safely navigate through shallow areas [18]. While the waterway provides a clear path for navigation, the depth and width of the waterway can vary significantly, as shown in figure 2.



2 Depth of a Waterway

If the tide is high, vessels may be able to sail further along the edge of the waterway, as a minimum depth is still ensured there. This may not be the case at low tide, so vessels will have to sail in the middle of the waterway. In general, a vessel has to navigate as far to the right as possible within the waterway, but depending on the tide and other factors, this distance can vary for each vessel [19].

3 Machine Learning

This chapter focuses on the background of time series prediction using machine learning. In particular, this chapter explains recurrent neural networks (RNN), Long Short-Term Memories (LSTM) cells and variations of architectures that can be used to predict time series. In addition, this chapter covers a transformer model that processes data differently than those previously mentioned, but can also be used to predict time series.

3.1 Recurrent Neural Networks

In the field of neural networks, there are two basic architectures that differ in how the neurons are connected to each other. These architectures are called feed-forward networks (FFN) and recurrent neural networks (RNN). Both types of networks consist of neurons that are used as processing units. These neurons are interconnected within the network and receive weighted information from the previous neurons, information passed on to the next neurons via an activation function. Activation functions are needed to add nonlinearity to a model. In an FFN, the information flows only from the input to the output in one direction, without feedback connections. These networks are well-suited for tasks where the input and output are independent, such as image classification or speech recognition. [20]

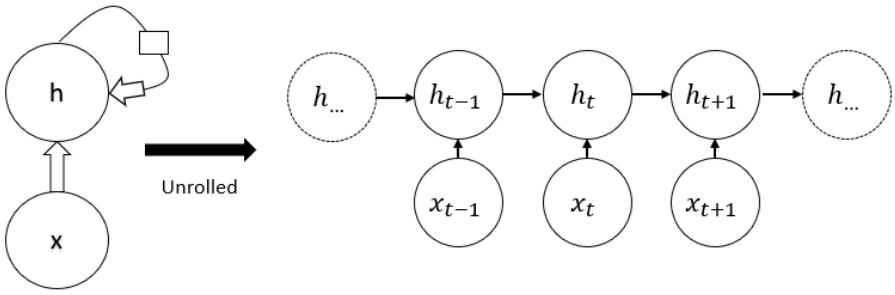
In order to process sequential data, such as time-series data, recurrent neural networks show more accurate results than feed forward networks. These networks were developed by Rumelhart et al. [21] and process an input sequence x one after the other and thereby update an internal state h that is used to predict the output sequence y. Accordingly, the internal state h at timestep t is calculated using the input x at timestep t and the previous internal state h_{t-1}. This can be represented in equation 1:

$$h_t = f(h_{t-1}, x_t) \quad \textbf{1}$$

This recursive transition can be unfolded for a finite amount of timesteps t, for example $t = 3$, where h_0 is a predefined initial state:

$$h_3 = f(h_2; x_3) = f(f(h_1; x_2); x_3) = f(f(f(h_0; x_1); x_2); x_3) \quad \textbf{2}$$

This unrolled computation can be seen in figure 3, which also shows how the internal state gets updated depending on the previous state h_{t-1} as well as the current input to the network x_t.



3 Recurrent Neural Network [22]

One reason these networks are good for processing sequential data is that the transition function f can be used with the same parameters for all time steps, since the input data size is the same for all steps. This avoids the need to learn new parameters for each time step, which reduces training time and requires less data. Goodfellow et al. [22] explain this by representing the unrolled recurrence after t timesteps with a function g_t.

$$h_t = g_t(x_t, x_{t-1}, x_{t-2}, \dots, x_2, x_1) \quad \textbf{3}$$

$$h_t = f\left(h_{(t-1)}; x_t\right) \quad \textbf{4}$$

The function g_t takes the whole past sequence $(x_t, x_{t-1}, x_{t-2}, \dots, x_2, x_1)$ as an input and can be factorized into repeated applications of a function f due to the recursive structure. This shows that f can be applied to all time steps.

During the forward propagation inside an RNN, an output sequence y is predicted that is based on an input sequence x. Based on the parameter sharing just introduced, this can be represented as follows: b and c are biased vectors which are applied before each activation function while W is a weighted matrix for hidden-to-hidden connections, U for input-to-hidden and V for hidden-to-output connections. These vectors and matrices are adjusted during training to improve the prediction result. During the first step, the previous state $h_{(t-1)}$ and the current time step x_t are added together with the corresponding matrices.

$$a_t = b + Wh_{t-1} + Ux_t \quad \textbf{5}$$

Then the hidden state h_t is calculated using an activation function, e.g. the hyperbolic tangent function $tanh$, which maps the input to a value between -1 and 1.

$$h_t = tanh(a_t) \quad \textbf{6}$$

The hidden state is then combined with the associated weighted matrix and finally the output is generated using another activation function, such as the softmax function.

$$o_t = c + Vh_t \quad \textbf{7}$$

$$\hat{y}_t = softmax(o_t) \quad \textbf{8}$$

The total loss L for a given sequence compared to the ground truth sequence y would then be the sum of the losses over all time steps, e.g. calculated with the Mean Square Error (MSE).

$$L(\{Y_1, \ldots, Y_t\}, \{\hat{Y}_1, \ldots, \hat{Y}_t\}) = \frac{1}{t}\sum_{t=1}^{t}(Y_t - \hat{Y}_t) \quad \textbf{9}$$

These loss functions are an essential part of neural networks since they quantify the difference between the predicted output and the actual output of the network. The loss can be measured with different functions. For regression problems, where the goal is to predict a continuous value, the MSE or the Mean Absolute Error (MAE) are often used. The MSE measures the average squared difference while the MAE measures the average absolute difference between the predicted and actual values.

During training the model updates the weights through an optimizer and backpropagation. These optimizers are responsible for updating the model parameters during training so that the loss function, which measures the difference between the predicted output and the actual output, is updated. In this process, the gradient of the loss function is recursively calculated for the nodes within the network by multiplying the gradients of the previous layer with the weight matrix that connects the two layers. This calculation is done throughout the whole network which is called backpropagation [20].

There are various optimization algorithms and the choice of the optimizer depends on the specific problem and the characteristics of the data on which the model should be trained. One simple optimizer is the Stochastic Gradient Descent (SDG). This optimizer updates the model parameters in the direction of the negative gradient of the loss function. However, this optimizer can be slow and ineffective. Another optimizer is the Adaptive Moment Estimation (Adam) optimizer. This optimizer allows handling different scales and directions of gradients for each parameter. It further includes a momentum-based update that incorporates information from past gradients to stabilize the optimization process. Specifically, the Adam optimizer maintains a running average of the first and second moments of the gradient, which are respectively the mean and variance of the gradient values. These estimates are used to compute a per-parameter learning rate that adapts to the gradient history. The algorithm also includes a momentum term, which allows the optimizer to move more confidently in the direction of the gradient. [23]

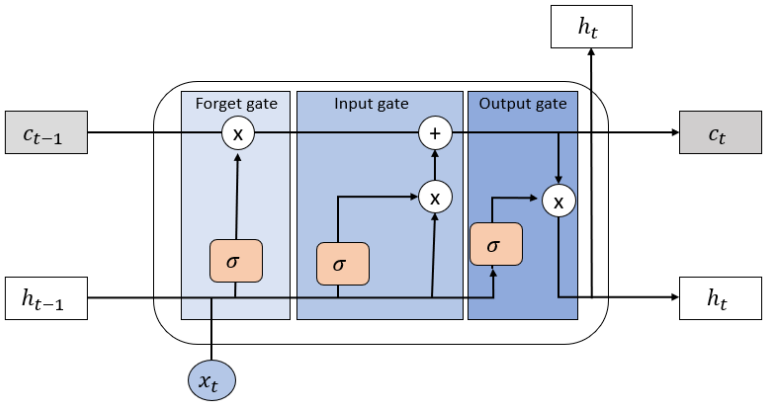
Particularly for deep RNNs, where the gradients can be multiplied by the same weighting matrix at each time step, the gradients can become very small or even zero when the weighting matrix is less than 1. This occurs especially in deep networks with many layers and leads to the vanishing gradient problem, where the gradients become too small to provide a useful signal for updating the weights [24].

There are several techniques that can be used to mitigate the vanishing gradient problem. One approach uses activation functions that have larger derivatives [25]. Further initialization methods that can make it easier for the gradients to propagate can be used [22]. Another approach is using gradient clipping to prevent the gradients from becoming too large or too small [26], or using a different type of network architecture that is less susceptible to the problem.

3.2 Long-Short Term Memory

To address the vanishing gradient problem and increase the ability to learn long-time dependencies, Hochreiter et al. [27] developed Long Short-Term memory (LSTM) cells. In doing so, they introduced the idea of a cell state that acts like a self-loop, allowing the gradient to flow over long periods of time. Gers et al. [28] then made these cell state weights trainable, leading to the widespread use of this architecture. This architecture has the same inputs and outputs as an RNN but adds an additional cell state that is not passed to other cells. Instead, it controls the flow of information within the LSTM.

Figure 4 shows a LSTM cell. The input x_t and the hidden state h_{t-1} and the cell state c_{t-1} enters the cell. From thereon they pass through three different gates, a forget gate, an input gate and an output gate.



4 Long Short-Term Memory Cell [29]

The forget gate controls the internal cell state, which is set to a value between 0 and 1 using the sigmoid function σ for the current input x_t and the previous hidden state h_{t-1}.

The output of the previous cell state c_{t-1} is then multiplied by the output of the forget gate. If the forget gate is 0, it means that the cell state should be forgotten, if it is 1, the cell state should remain unchanged. W, U are again trainable weight matrices and b, j are biased vectors.

$$f_t = \sigma\left(b_f + W_f x_t + U_f h_{t-1}\right) \quad \textbf{10}$$

$$w_t = f_t \odot c_{t-1} \quad \textbf{11}$$

The input gate then calculates which of the new information should be added to the cell state. The new cell state candidate $\hat{c}_t$ is calculated by activating the input x_t and the previous hidden state h_{t-1} in a hyperbolic tangent function $tanh$. As with the forget gate, the previous hidden state h_{t-1} and the current input x_t are fed into a softmax function σ that decides whether the cell state should be updated or not. The new internal cell state c_t is then calculated by combining the output of the forget gate with the output of the input gate.

$$i_t = \sigma(b_i + W_i x_t + U_i h_{t-1}) \quad \textbf{12}$$

$$\hat{c}_t = tanh(b_s + W_s x_t + U_s h_{t-1}) \quad \textbf{13}$$

$$c_t = w_t + i_t \odot \hat{c}_t \quad \textbf{14}$$

Finally, the output gate takes the previous hidden state h_{t-1} and the current input x, again activated through the softmax function σ, as well as the hyperbolic tangent function of the current cell c_t state and updates the new hidden state h_t that should be forwarded to the next cell.

$$o_t = \sigma(b_o + W_o x_t + U_o h_{t-1}) \quad \textbf{15}$$

$$h_t = o_t \odot tanh(c_t) \quad \textbf{16}$$

LSTMs have led to significantly improved results compared to the native RNNs [22]. Graves et al. [30] e.g. used LSTMs to develop a model to recognize unconstrained handwriting, Vinyals et al. [31] for image captioning or for predicting time series, e. g. in the case of vessel tracks by Zhang et al [32], which is also the use case in this thesis.

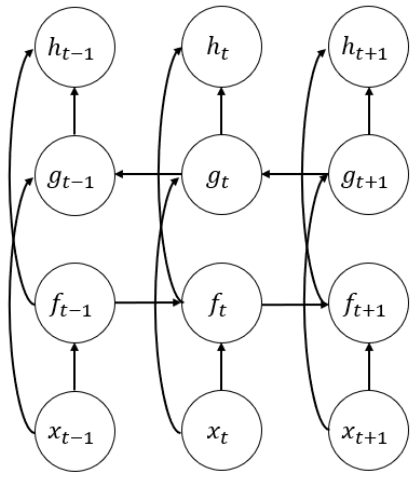
3.3 Variations of RNNs

As mentioned in the previous section, the original RNN as well as the adjusted LSTM are especially suitable for processing time series data. However, these have been refined over time and different architectures of networks have emerged that provide

better results depending on the use case [22]. Nevertheless, the basic structure can always be traced back to an original RNN as well as an LSTM architecture. Accordingly, the recurrent neural networks shown in the following chapter can be used for both classical RNNs and LSTMs models.

Bi-directional RNN

A bidirectional RNN, as introduced by Schuster et al. [33], processes input data in both forward and backward directions. Therefore, they combine two separate RNNs. One RNN moves forward in time, considering the incoming sequence from the beginning, while another RNN moves backward, considering the sequence from the end. This process is shown graphically in figure 5. The outputs of the forward and backward RNNs are then concatenated to produce the final output of the bidirectional RNN. This makes the output not only a summary of the past but also on the future sequence and thus may make it easier to recognize certain patterns.

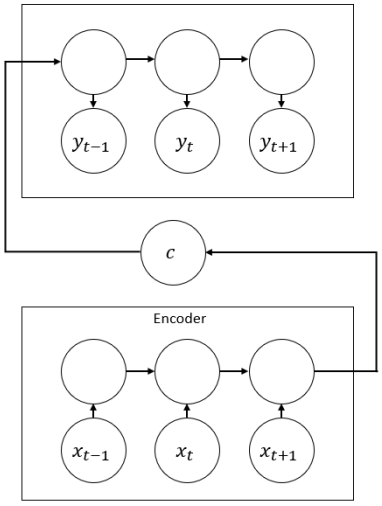


5 Bidirectional RNN where x displays the input at time step t where f process the input forward in time and g backwards. Hidden state h than summarizes the information from the two neurons.

Sequence-to-Sequence RNN

A sequence-to-sequence or encoder-decoder RNN is designed to handle variable-length input and output sequences. Thereby the RNN consist of two RNNs, an encoder and a decoder. As Figure 6 shows, the encoder takes the input sequence and produces a fixed length context vector that stores all the information of the input. The decoder then takes

the context vector and produces an output sequence. Depending on the architecture, the decoder can get the context vector as an initial state or the vector can be connected to the hidden units of the decoder at each time step. [34]



6 Sequence-to-Sequence RNN where input sequence x is summarized to context vector c and then fed to decoder as an initial state that produces output y.

3.4 Attention Mechanism

The introduced fixed-size context vector in sequence-to-sequence RNNs has to store the entire information produced by the encoder. Since this vector only has a certain size, the information may have to be greatly compressed and consequently not store all relevant information [22]. To solve this problem, Bahdanau et al. [35] proposed a method that encodes the input sequence into a variable-length context vector. They further introduced an attention mechanism that determines a relevant subset of the context vector to generate an output.

Unlike the original encoder-decoder model, where the hidden state was passed from one layer to another, the attention mechanism creates a hidden state h for every input at time step i. Since the encoder is a bidirectional RNN that combines both the preceding and following values within the sequence, each hidden state h is concentrated on the parts surrounding the i-th step. Accordingly, the context vector c at time step i is calculated as the weighted sum of all these hidden states.

$$c_i = \sum_{j=1}^{T} a_{ij} h_j \quad \textbf{17}$$

Where a_{ij} is the weight of each hidden state h_j that is calculated with an alignment score scoring how well the inputs around position j and the output at position i match. As an alignment function, Bahdanau et al. [36] used the hyperbolic tangent activation, which is also called additive attention.

$$score_{ij} = align(s_{i-1}, h_j) \quad \textbf{18}$$

This score is then normalized using the softmax function σ.

$$a_{ij} = \sigma(score_{ij}) \quad \textbf{19}$$

This attention mechanism was adopted for many applications. Thus, the attention mechanism was used as a context vector between the encoder and the decoder, since it is better able to store the information provided by the encoder which is then passed on to the decoder [32].

3.5 Transformer

In 2017, Vaswani et al. [37] developed a novel architecture that is entirely devoid of RNNs and focuses only on attention mechanisms.

For this they used a concept called "Self-Attention", which is based on the attention mechanism concept explained in the previous section but is used within a single input sequence. In their concept, three different vectors, called query Q, key K, and value V vectors, are created by multiplying the input with three corresponding weight matrices. These three vectors are then used to calculate the attention scores for the input sequence. For this, Vaswani et al. [37] used "Scaled-Dot-Product-Attention" where the dot product of query Q and key K is calculated. This produces a score that indicates the relevant parts of the sequence that should be focused on. These scores are then scaled with a scaling factor $\sqrt{d_{(k)}}$ and normalized through a softmax function σ to find the most relevant inputs with respect to the whole sequence. The result of the softmax function is then multiplied by the value matrix V to filter out irrelevant inputs.

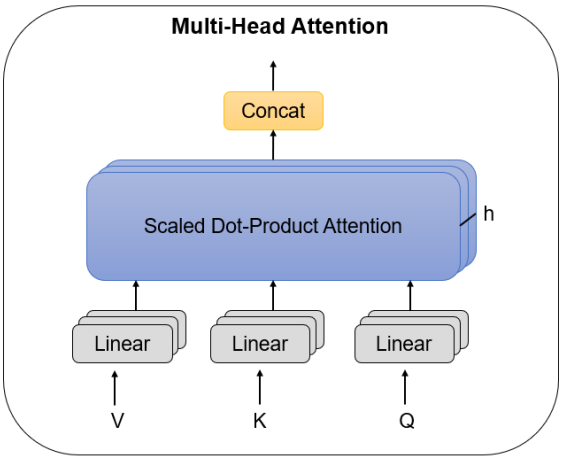
$$attention(Q, K, V) = \sigma\left(\frac{QK^t}{\sqrt{d_k}}\right)V \quad \textbf{20}$$

The attention mechanism is then scaled by using multiple sets of queries Q, keys K, and value V matrices, referred to as "multi-headed" attention.

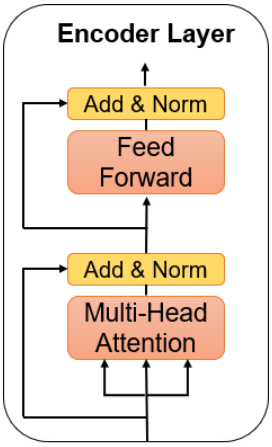
$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W_{out},$$
$$with\ head_i = attention(QW_Q^i, KW_K^i, VW_V^i) \quad \textbf{21}$$

This should allow each attention head to focus on a different position within the input sequence. These different heads are then concatenated and multiplied by an associated weight matrix W_{out} so that only the relevant information within the input sequence is captured and forwarded.

Figure 7 summarizes the explained process of Multi-Head Attention.



7 Multi-Head Attention



8 Transformer Encoder Layer

The vectors Q, K, and V, which result from the input sequence, where multiplied with three corresponding weight matrices. Further, scaled dot product attention is used to calculate the attention score to indicate the relevant parts of the sequence that should be focused on. The attention mechanism can be scaled by combining several heads h, which allows encoding multiple relationships for each input. All produced attention scores are then combined to produce a final attention score.

In the model presented by Vaswani et al. [37], the multi-head attention mechanism is followed by a normalization layer and a fully connected feedforward network. As can be seen in Figure 8, skip connections inside the encoder improve the flow of information through the network and the flow of gradients during backpropagation. These layers can be stacked on top of each other, with the input to the first layer being the input sequence and the output to subsequent layers being the output of the first encoder layer.

The decoder also consists of several identical layers. It receives the output of the encoder, which is converted into a set of keys and values. These are then combined with the output of the decoder from the previous step.

A common problem that can occur when training deep neural networks is the so-called overfitting. Overfitting is a problem that arises when a model is too complex and starts to memorize the training data instead of generalizing to new data. To prevent overfitting of the model, Vaswani et al. [37] used a regularization technique called dropout. Specifically, dropout is applied to the attention weights and the output of the feed-forward network within each sub-layer. Dropout was introduced by Srivastava et al. [38] and is

widely used in machine learning to prevent overfitting. It is a simple yet powerful technique that has been shown to improve the generalization performance of deep learning models. Dropout works by randomly dropping out neurons during training. Thereby, it forces the remaining neurons to take on more useful and diverse roles, which improves the generalization performance of the model. When a neuron is dropped out, it is effectively removed from the network and its input and output connections are temporarily removed. The remaining neurons in the layer must then learn to compensate for the missing neuron, which forces them to take on more useful and diverse roles. This dropout technique is only used during training. During predictions, all neurons are present but their outputs are scaled by a factor of the dropout rate to compensate for the dropout during training. This ensures that the expected output of each neuron is the same during training and prediction.

Unlike RNN networks, transformers map the input at a specific time step against all keys and consequently do not remember the order of the input tokens. To address this issue, transformers use positional encoding to provide the model with information about the position of each token in the input sequence. For this, Vaswani et al. [37] applied a sinusoidal function that can be summed to the input sequence where pos is the position in the sequence, i the index of the dimension within the positional encoding and d_{model} the dimensionality of the model:

$$PE_{(pos,2i)} = sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \quad \textbf{22}$$

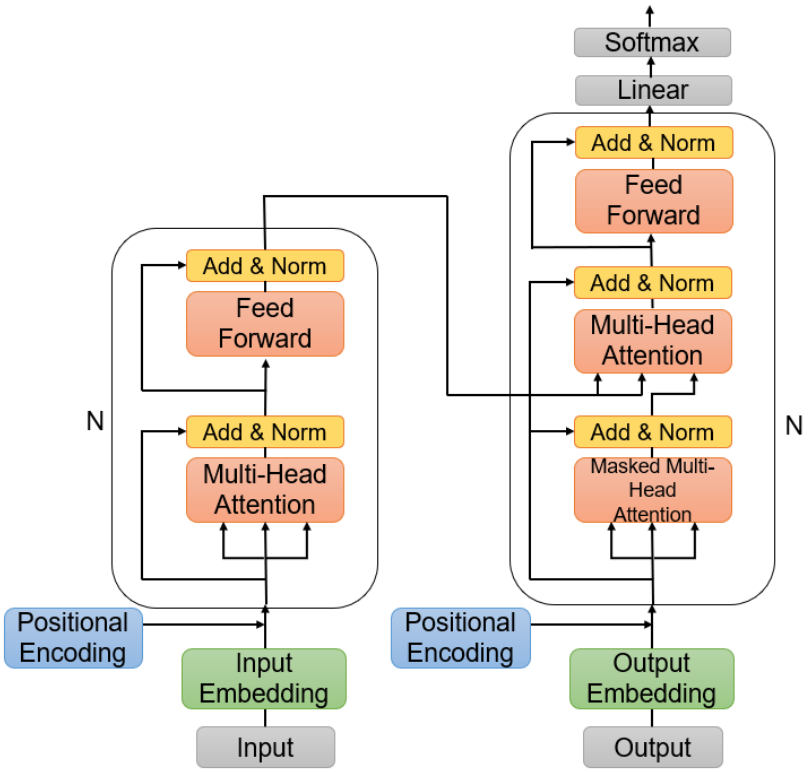
$$PE_{(pos,2i+1)} = cos(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \quad \textbf{23}$$

During training, the network will then learn how this information can be utilized. In addition to this approach, other ways of positional encoding can be used. Kazemi et al. [39], for example, introduced a method called Time2Vec (t2v) that was especially developed for time series data. This method applies a learnable function to the time index so that it can capture more complex patterns, such as seasonality and periodicity, where l stands for the linear part and p for the periodical part of the method.:

$$t2v_l = w_l x + b_l \quad \textbf{24}$$

$$t2v_p = sin(w_p x + b_p) \quad \textbf{25}$$

Figure 9 visualizes the whole transformer structure introduced by Vaswani et al. [37]. It is built like a sequence-to-sequence network, with the RNN replaced by attention. The authors stacked 6 encoders and 6 decoders and used it for translation tasks, where it outperformed the best previously reported models.



9 Transformer Architecture [37]

As mentioned before, the transformer model was originally introduced for the task of machine translation, where the goal is to translate a source sequence (e.g., a sentence in one language) into a target sequence (e.g., a sentence in another language). In this task the decoder part of the transformer model is used to generate the output sequence one token at a time, while attending to the relevant parts of the input sequence. However, in the case of time series forecasting, the task is different. A given sequence of past values should be used to predict future values of the same sequence. Therefore, only the encoder part of the model can be sufficient. This part extracts features and captures temporal patterns from the input sequence that are relevant for predicting the future values [40]. The features extracted by the transformer model can then be fed into a simple feedforward neural network to predict the future values.

When predicting long time series, Zhou et al. [41] pointed out that the self-attention mechanism, as well as the stack of encoder/decoder layers inside the original transformer architecture, are not computationally efficient. Therefore, the authors introduced a ProbSparse self-attention mechanism, in which the keys are only combined with dominate queries which are learned during training. Accordingly, the computational costs are reduced while maintaining the self-attention idea. Furthermore, they introduced Informer, a generative style decoder that contains the masked multi-head attention

mechanism from the original paper as well as the ProbSparse attention mechanism, which is used to select the most relevant historical values to predict the future ones. As results show, the Informer architecture outperforms the original transformer as well as other LSTM models for long time series forecasting.

Tailored to multivariate time series, Zerveas et al. [42] proposed a method to process the data in a more efficient way. While the initial approach considers the input sequence as a whole, their method considers each attribute within the sequence. Zerveas et al. [42] order the input sequence in a way that makes each token contain a single variable at a given timestep. Since the computational power for matrix multiplication increases with the consideration of each attribute, a different kind of attention mechanism, namely performance attention, is used. Thereby the attention mechanism is calculated using random feature approximations instead of the exact softmax computation used in the original transformer architecture. As a result, Zerveas et al. [42] show that their model increases the accuracy in multivariate classification and lessens regression problems.

4 Related Work

In this chapter, previous research on vessel track prediction and anomaly detection in the maritime domain is presented and contextualized.

4.1 Vessel Track Prediction

Vessel trajectory prediction has been intensively researched in recent years. Zhang et al. [8] summarize 57 different papers dealing with research in this field. The prediction methods can be distinguished between statistical and machine learning methods, allowing methods to also be combined with each other. However, Zhang et al. [8] state that the area of machine learning has become increasingly important since the year 2020. Accordingly, this work focuses predominantly on machine learning methods while also considering select hybrid or statistical methods relevant for this thesis. State-of-the-art methods for predicting vessel trajectories include LSTM cells. These methods are often extended with other methods from the field of machine learning. An overview of the methods used in the research surrounding vessel trajectory prediction is given in table 4, sorted by machine learning or statistical methods, grouping research findings by method and not necessarily by time.

In the approach introduced by Mehri et al. [43], separate models for each type of vessel were constructed using LSTMs. To train these models, AIS data were used from November 2017 to the end of December 2017 from the eastern coast of the United States of America. In addition, geographic information was used to simplify vessel tracks. The models were then evaluated using the Root-Mean-Square Error (RMSE) and the point-wise horizontal error and compared with an ordinary LSTM model. The results showed that the developed method receives a lower point-wise error in predicting the tracks accuracy with a range of up to 2 kilometers than an ordinary LSTM model.

Authors	Method	
	Machine Learning	Statistical
Mehri et al. [43]	LSTM	
Gao et al. [44]	MP-LSTM	TPNet
Yang et al. [45]	Bi-LSTM	
Liu et al. [46]	Bi-LSTM + attention	
Zhang et al. [32]	Bi-LSTM + attention	
Venskus et al. [47]	Autoencoder LSTM	
Nguyen et al. [48]	Seq-2-seq LSTM	
Dijt et al. [49]	Encoder-decoder LSTM + CNN	
Forti et al. [50]	Encoder-decoder LSTM	
Capobianco et al. [51]	LSTM + attention	
Sekhon et al. [52]	LSTM + attention	
Ding et al. [53]	Variational LSTM	
You et al. [54]	Seq-2-seq GRU	
Nguyen et al. [4]	Transformer	
Steidel et al. [18]		KDE

4 Overview Methods Vessel Track Prediction

Gao et al. [44] further developed a hybrid method including statistical and deep learning methods. For this, they combined the TPNet framework and LSTM cells. TPNet is used to predict vehicle trajectories by modeling them as polynomial continuous curves with the help of an end point and a curvature point. The authors mention that for predicting vessel trajectories, this model has certain issues, partly because TPNet handles predictions as classification rather than regression tasks. However, predictions as regression task should allow a higher accuracy to be achieved. Accordingly, the authors

use an LSTM model to predict the curvature point and consider the prediction as a regression problem. The endpoint is predicted based on the hypothesis that the values in the generated and current trajectory are approximately equal. In the final step, the trajectory is simulated by the cubic spline interpolation based on the predicted curvature and end point. This model was tested with 1000 tracks of ferry vessels sailing along the Jiangsu section of the Yangtze River. It was then evaluated against several other models by comparing various navigation phases, such as straight-line phase and turning phase. Among the models that were compared, there are both LSTMs that do the prediction and LSTMs that predict the changes of latitude and longitude and calculate the new position based on these changes. They also evaluated the model against a dual linear autoencoder proposed by Murray et al. [55]. The authors show that the developed MP-LSTM has both the smallest final displacement error and the smallest average displacement error among the evaluated models. Furthermore, they state that the model works for both long and short distance predictions.

Yang et al. [45] on the other hand refined an LSTM by designing a Bi-LSTM model where one part learns dependencies for the states following and the other part for the previous states. The combination of these two predictions should then lead to a more accurate prediction of the following states. The model was trained with AIS data from 1364 ship tracks collected during one month in the Taiwan area. It was then evaluated by comparing the MAE, MSE and the Mean-Absolute-Percentage Error (MAPE) with various other methods, such as simple LSTM and RNN models. For this, they randomly compared 10 vessel tracks. The results showed that the BI-LSTM has the lowest error in all categories compared to the other models. Based on this, the authors state that the developed model can accurately predict short-term trajectories.

A Bi-LSTM model can also be found in the work by Liu et al. [56]. They developed a series of routing algorithms where a Bi-LSTM was augmented with an attention mechanism to predict the next position of a vessel along the trajectory. The attention mechanism should make the prediction more accurate by better learning the dependencies of the AIS data. This method was trained with AIS data from fishery vessels along the east coast of China from May 2015 to May 2018. The results showed that the methods predict vessel positions to an error of less than 300 meters after one hour and an error of 2.73 kilometers after 9 hours when using an iterative process.

Zhang et al. [32] used an auto-encoder LSTM structure to extract trajectory features in a first step and in a second step combined an attention-based Bi-LSTM model with dropout layers they call AABIL to predict vessel trajectories. To train the model, AIS data from the U.S. East Coast from January 2017 was used. Their results showed that the MSE of

the constructed model is lower compared to an ordinary LSTM model and they concluded that a combination of a bidirectional LSTM and an attention mechanism can improve the accuracy of real-time trajectory predictions.

An autoencoder LSTM model as well as a wild bootstrapping technique was developed by Venskus et a. [47]. They trained an autoencoder LSTM to predict a region the vessel should be in after up to 2.4 hours. This approach was then evaluated using the MSE against a wild bootstrapping technique. To train and evaluate this method they used AIS data from cargo ships along the Danish waters from 2006 to 2020 as well as meteorological data from 2019 to 2020. Results showed that the LSTM learns a wider possible region than the wild bootstrapping technique.

A slightly different approach for trajectory predicting, but also with the use of a sequence-to-sequence LSTM, was developed by Nguyen et al. [48]. They divided the Mediterranean Sea into grids of 1 x 1 nautical miles and trained a sequence-to-sequence LSTM model that predicts the trajectory by predicting the following grids that the ship will approach. The model was then trained with AIS data from March to May 2015 and valuated against a Gated Recurrent Unit (GRU) and other networks. Results showed that the LSTM shows the lowest log perplexity and is therefore well suited for predicting ship trajectories, as the authors note.

A complex model was developed by Dijt et al. [49], who used AIS data as well as RADAR and Electronic Navigational Charts (ENC) data to train a model and predict the trajectories of vessels for up to 6 hours. For this, they developed a model consisting of three modules. The convolutional module combines the RADAR data with the coordinates from the AIS data using Convolutional Neural Networks (CNN). The second module is a recurrent module which consist of an encoder-decoder LSTM model that predicts the future coordinates of the trajectories. Lastly, a segmentation module is used which outputs the results as binary string. This model is then trained with de MSE and the Euclidean distance as error. The authors state that the developed method is capable of predicting long-term trajectories of inland vessels and show that the developed model has a lower error than other state-of-the-art models used for trajectory predictions of pedestrians.

Another track prediction method based on an encoder-decoder LSTM architecture was proposed by Forti et al. [50]. In this method, the encoder consists of 64 LSTM cells and the decoder of 32 cells. The performance was evaluated so that the model was given five and 20 steps as input sequences and the next 20 output sequences were to be predicted, sampling the timestamps at two-minute intervals. The data were collected

from June to September 2018 and include 534 different voyages from the Port of Piombino to Portoferraio on Elba Island, Italy. The authors performed a qualitative comparison based on a stochastic Ornstein-Uhlenbeck (OU) model developed by Millefiori et al. [57] by calculating the RMSE. The authors state that the encoder-decoder LSTM method achieves competitive performance compared to the OU method, especially near waypoints.

Based on this approach, Capobianco et al. [51] also developed an encoder-decoder LSTM. However, they choose a more complex structure in which the encoder is constructed as a Bi-LSTM and the decoder contains LSTM cells and multi-layer perceptron that convert the input into an output sequence. Further, they investigated three different approaches to compressing the information generated by the encoder: maximum pooling over time, average pooling, and an attentional mechanism. These methods were then evaluated with both unlabeled and labeled trajectories that additionally contained the target area. All methods were trained with AIS data from January to February 2020 provided by the Danish Maritime Authority (DMA) and included only tanker vessels. The authors then divided the data into input and output sequences of fixed lengths of up to three hours. Results show that the attention-based aggregation function leads to more accurate results and that performance can be improved by adding information about the vessel destination.

Sekhon et al. [52] also developed a method for short-term predictions of ship tracks using LSTM with spatial and temporal attention. In addition to an LSTM model, their encoder uses spatial attention to extract information from neighboring vessels that may affect the trajectory of the vessel under consideration. The decoder then uses both spatial and temporal attention to learn the important parts of the encoded vector. The final prediction is also made using LSTMs. The model was trained with AIS data from January 2017 around the Port of San Diego, USA. The results show that both ADE and FDE are lower when using the LSTM in combination with spatial and temporal attention compared to methods using only one of the two attention mechanisms as well as an original LSTM architecture.

Again a different approach was taken by Ding et al [53], who created a variational LSTM with the idea that it has a stronger learning capacity and is better able to extract features from trajectory data than an original LSTM model. It was trained with 2017 AIS data from the U.S. West Coast and then compared to an original LSTM model. To evaluate the model, the next five to 20 time steps were predicted, with each time step greater than one minute after the other. The results measured with the MSE show that the variable

LSTM is more accurate in predicting the future trajectories, especially in predicting later time steps.

Instead of an LSTM, You et al. [54] developed an encoder-decoder GRU model that converges faster than an LSTM model due to the smaller number of cells. The authors trained the model using AIS data from the first 10 days in June 2017 from the Whuan Waterway and from the Chongquing Waterway and evaluated the model against an LSTM and a GRU model using the RMSE. They used a ten-minute time interval to predict the following five minutes of the vessel trajectory and results showed that the sequence-to-sequence GRU predicts the lowest RMSE.

Nguyen et al. [7], in turn, place all the focus on the attention mechanism by using a transformer deep learning model for vessel trajectory prediction. They state that standard deterministic approaches such as LSTM or GRU cannot capture the multi-model patterns involved in AIS data and are therefore ineffective for trajectory prediction. With this assumption in mind, Nguyen [4] proposed a transformer structure to deal with the multi-model nature of AIS tracks.

In contrast to Gao et al. [44], the prediction by Nguyen [4] is considered as a classification-based problem rather than a regression problem. Therefore, the AIS data with attributes latitude, longitude, speed-over-ground and course-over-ground are discretized into a higher dimensional vector containing all attributes. This transformer model contains 8 layers, each with 8 attention heads, and was trained and tested with AIS data along the Danish coast during the first three months of 2019. The model was then evaluated, among others, against the sequence-to-sequence LSTM model proposed by Forti et al. [50]. Results show a significantly lower error in the forecast, measured with the haversine distance, both in the first three hours and after 10 hours. The authors [4] state that the developed model is more suitable to capture the multi-model nature of AIS data and extract useful information from historical data than the previous models.

A statistical approach to predict vessel behavior, which is important for this work due to the preparation of the data, comes from Steidel et al. [18]. In addition to AIS data, they used buoys along the waterway to divide the vessel tracks into grids. Within these grids, behavior is predicted using Kernel Density Estimation (KDE). The dataset used by the authors consists of 653 cargo ship tracks on the German Elbe River. To evaluate the prediction performance, the ship domain of Fujii et al. [58] was used. The result was a prediction error of 85.99 meters measured for a 2.35-hour voyage.

In summary, the majority of research focusses on machine learning models using LSTM cells, which are being refined and specialized for the specific prediction tasks. Besides, there are other approaches that instead of using LSTM model, use GRU or transformers to improve the prediction accuracy.

The existing research just introduced can be divided further into different categories. The table 1 in the appendix divides the type of prediction into three categories: Single Step Prediction, Track Prediction, and Trajectory Prediction. Single Step Prediction means that the location of the next moment is predicted. It is also possible to predict multiple steps using an iterative approach. Track prediction, on the other hand, means that more than one future position is predicted at the same time. However, only the position is considered and not the relationship with time. The consideration of time is categorized into trajectory prediction, where the position and time can be predicted simultaneously.

As can becomes clear, most of the approaches predict vessel trajectories, while only some specialize in tracks and only Liu et al. [56] predict just the next position and then use an iterative approach to predict the following positions. Depending on the type of prediction, an iterative approach can provide more accurate predictions as the model needs to learn fewer complex dependencies. However, it must be kept in mind that predicted errors are also added to the model. This makes longer predictions rather less accurate. Although it is more complex to predict the trajectory, it is attempted by many researchers because the AIS data provided means the information need not be obtained further. Additionally, depending on the prediction target, time can also be an important characteristic, e.g. when an arrival time should be predicted.

Based on this, appendix 1 also classifies the time period in which the predictions take place between the different papers. In this thesis, predictions of up to one hour are classified as short, predictions of time spans between one and three hours are classified as medium, and predictions longer than three hours are classified as long. The distribution of the different prediction horizons is relatively balanced, although in general many models can give relatively accurate predictions for a short time, with these becoming less accurate the longer the movement is predicted.

The table in appendix 1 also classifies which data sources are used to train the models, dividing them into AIS, meteorological and geographical data. The geographic information is taken from nautical charts or boundaries to the shore. All approaches use AIS data to predict the vessel movement and only some supplement this data with meteorological or geographic data. AIS data are very often the basis for prediction tasks, as they enable the movements of a vessel to be tracked. However, other data sources

can be used to make the prediction more accurate. Tidal information, for example, can also have an impact on navigation, especially in coastal areas. Nevertheless, there is a lack of research that combines all three mentioned categories to predict vessel tracks. Steidel et al. [18] use AIS data as well as data from waterways but predicts tracks using KDE rather than a data-driven model. However, the information that is extracted from the waterways can also be useful when predicting vessel tracks, since it limits the space in which the vessel can sail. The information could also be attempted to be included in a data-driven model to predict where the vessel navigates, possibly capturing those dependencies. On top of that, tidal information and other weather data could be used to provide an even more accurate prediction.

The combination of historical vessel data, tide data, and weather information, which has not yet been sufficiently considered in research, is incorporated in this thesis. In this process, a data-driven model is trained to predict a vessel's next positions within a waterway. To create a dataset, AIS data as well as data from waterways will be used. Furthermore, it will be examined to what extent tide and weather information can have an influence on the prediction. For this purpose, these data will also be added to the dataset. The model will be trained on this data and the results will be evaluated. In this thesis, an iterative approach is used to predict the next positions inside the waterway. This makes the prediction of the model slightly less complex and should lead to more precise results. The selected approach to predicting vessel tracks fits into appendix 1 in such a way that it predicts tracks iteratively, using all three combinations of data while making prediction for a short horizon. Accordingly, the chosen approach complements the research in a novel manner.

4.2 Anomaly Detection

Data-driven approaches to anomaly detection of vessel behavior often consist of two parts. In the first part, normal vessel behavior is learned from historical data and then in the second part, it is compared to other data to detect anomalies. An anomaly thereby indicates that a certain threshold value has been exceeded. Accordingly, many methods developed for vessel trajectory prediction also mention that they can be used to detect abnormal vessel behavior.

Steidel et al. [18] for example mention that their approach to predict vessel behavior using KDE can also be used for anomaly detection. By dividing a waterway into regions where the vessel would normally travel, it could be used to detect when the vessel is moving outside these regions or on the opposite lane inside the waterway.

The autoencoder LSTM of Venskus et al. [47] described in the above section can also be used to detect abnormal vessel behavior. As summarized, in this approach a prediction region is learned for the vessel and if the actual position of the vessel is outside this region it is classified as abnormal vessel behavior, according to the authors.

Besides these approaches, there are other approaches that specify on the detection of anomalies in the maritime domain. One approach for this was developed by Ristic et al. [59]. Here, anomalies can be detected based on specific positions as well as the speed of the vessel. To accomplish this, the area of interest is divided into cells where vessels normally travel. Anomalies are then detected by comparing the position and speed with the learned distribution. The method was tested with AIS data from January to May 2009 in the Port of Jackson, Australia.

A probabilistic normalcy model of vessel dynamics was created by Guillarme et al. [60] using unsupervised learning techniques. The authors first partitioned the trajectories, then clustered these trajectories and finally used the clustered trajectories for modeling paths alongside which the vessels would usually continue. Using the extracted paths, the model checks whether new observations fit into the modeled paths and triggers an anomaly if they do not. Therefore, static AIS data is also used to compare the specified destination with the detected position. For training the model, AIS data from the Mediterranean Sea and the Straight of Gibraltar was used.

Another approach was developed by Lei et al. [61], who introduced a data-driven model that extracts behavioral features from trajectory data. This is done by converting raw AIS data into region-based motion trajectories. The authors then developed a model that detects anomalies when a given trajectory deviates from the learned behavior by more than a certain threshold. The model was tested with three months of AIS data and with anomalous tracks added randomly.

Vespe et al. [62] note that approaches which divide the area into grids, like Lei et al.'s [61], require a sufficiently small cell size and are therefore insufficient for global applications. They therefore developed an unsupervised approach to learning movement patterns, such as turning points, by analyzing AIS data. To do this, they used a three-month period of AIS data from the Adriatic Sea and the Red Sea and Gulf of Aden. The model can then be used to detect vessels that do not move according to the learned patterns. The authors further suggest that the approach can be used to support other surveillance technologies such as vessel position prediction.

One approach that is frequently referred to when predicting anomalies is called Density-Based Algorithm for Discovering Clusters (DBSCAN) by Ester et al. [63]. With this

approach, clusters of arbitrary shape can be discovered in an efficient way. To achieve this, a neighborhood around each datapoint is defined from which dense regions of points are identified as clusters. Based on this approach and applied to the maritime sector, Pallotta et al. [64] developed an unsupervised methodology to incrementally extract information from AIS data and detect low-likelihood vessel behavior. This approach, called Route Extraction for Anomaly Detection (TREAD), was then further developed by the authors to detect whether a vessel is off-route, in reverse traffic on the route or whether the speed is not compatible with the route followed [65]. They used AIS data in the Ligurian Sea from January to February 2013 and extracted this data into tracks and then compared these tracks to the extracted routes. An anomaly is flagged when a threshold value of the route is exceeded. The results show that 87.3 percent of the routes are correctly classified as anomalous. The authors further state that the ability to detect anomalies is highly dependent on how regular the traffic patterns in the observed area normally are.

A slightly different approach comes from Zhao et al. [66], who developed a neural network composed of LSTM cells to predict traffic patterns. The model further detects if the behavior of a vessel is abnormal by comparing the predicted position with the actual one. To train the model, patterns detected by the DBSCAN algorithm were used. The model was then tested with AIS data from the Zhoushan Islands dating from January 2015. As the authors explain, the capability of detecting anomalies with this approach depends heavily on the quality of the AIS data.

Table 5 summarizes the approaches listed. It can be seen that all methods rely on unsupervised clustering methods to learn the normality of the vessel trajectories. Additionally, all methods rely on AIS data to extract vessel tracks or to learn the traffic patterns. The approaches then differ in the method of detecting anomalies, although all recognize a deviation from the learned model above a certain threshold as an anomaly.

Authors	Learning Method
Ristic e al. [59]	Unsupervised clustering into grid cells
Guillarme et al. [60]	Clustering + path modeling
Lei et al. [61]	Region based movement pattern
Vespe et al. [62]	Movement patterns
Pallotta et al. [64]	TREAD
Pallotta et al. [65]	TREAD + Situational indicators
Zhao et al. [66]	DBSCAN + LSTM

5 Current Approaches Anomaly Detection

All but the approach introduced by Steidel et al. [18] can be used without extratcing information on the waterway which the vessel is navigating. However, the approaches introduced by Ristic et al. [59] or Guillarme et al. [60] also create clusters through which the vessel normally passes and then use data-driven approaches to detect deviations from the learned model. Accordingly, the information extracted from the waterway could also be used to create these clusters. These clusters would then be more precise, since only within the waterways a minimum depth would be given in which the vessel could navigate. Furthermore, instead of the KDE that is used by Steidel et al. [18], a data-driven model could be used to predict tracks within the waterways more accurately. Based on this model, deviations from the true track could be measured and if deviations are above a certain threshold, the track would be marked as abnormal. The usage of a data-driven model to detect anomalies within waterways may increase the usefulness of the prediction and would represent an advancement in research.

5 Concept

Having framed the research gap and the potential regarding data-driven models to predict and detect anomalies in vessel behavior above, this section presents a concept that combines multiple data sources to train data-driven models. In this concept, these data-driven models can then be used to predict not only vessel tracks iteratively within waterways but to also detect anomalous vessel tracks.

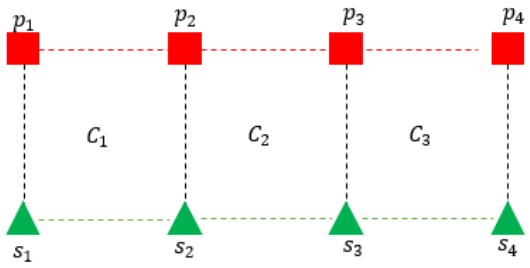
5.1 Vessel Track Prediction

In this thesis, a concept is developed to iteratively predict vessel tracks within waterways. To achieve this, different data features from AIS data as well as tide and weather data are be combined to obtain more accurate predictions. The prediction is based on AIS data. The considered AIS data attributes were described in table 6.

AIS Attribute	Description
Position	Latitude and Longitude coordinates in degrees are used to track the location of each vessel in real-time.
SOG	Speed Over Ground in knots provide information on the speed at which each vessel was moving
COG	Course Over Ground in degrees provide information on the direction in which each vessel was moving.
MMSI	This unique identification number assigned to each vessel allows for individual tracking and analysis of vessel behavior.
Time	The time stamp dates each AIS message

6 AIS Attributes Considered in Concept

In addition, the positions of the regions where waterways exist are extracted. It is assumed that a minimum depth within these waterways is given, that allows the vessels to navigate through them. Waterways contain starboard and port buoys that border the waterway. To find vessel tracks inside a waterway, the waterway can be divided into grids, where four buoys always form a cell. As it can be seen in figure 10, the two successive starboard buoys (s_1, s_2) and the two corresponding port buoys (p_1, p_2) always form a cell C_1. In general, this implies that C_n consist of the buoys s_n, s_{n+1}, p_n and p_{n+1}.

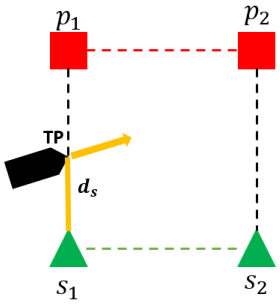


10 Waterway as Grid

In order to filter the AIS data within the waterways from all AIS data, the cells can be used. If the position of the AIS datum lies within a cell, the cell number is added to the AIS datum, otherwise the AIS datum is no longer considered.

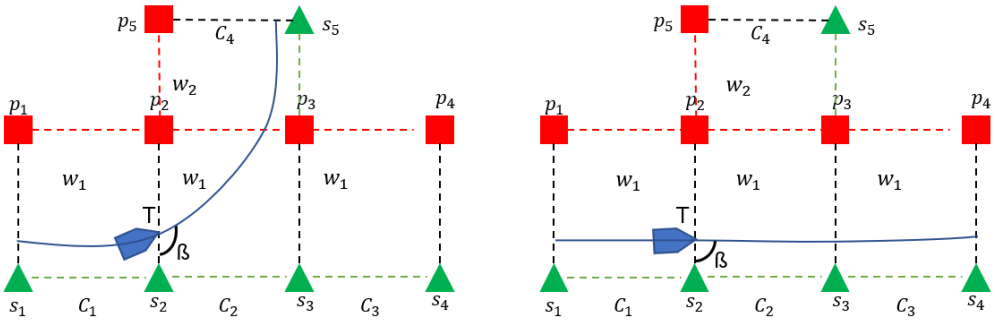
This filtered AIS data can now be used to create continuous AIS tracks along the considered waterways. An AIS track is defined as a series of AIS messages for a particular MMSI received within one minute of the previous message. If the interval between received messages is longer than one minute, a new track is created. Further, each track is marked with a track ID. This general procedure of extracting tracks is shown again as pseudo code in the appendix 2.

As mentioned in chapter 2.3, vessels inside a waterway are obliged to navigate as far right as possible to the starboard buoy. It is assumed that the distance of the vessel to the edge of the waterway remains constant within a cell. However, the distance to the starboard buoy may change, e.g. if the vessel navigates through curves. Additionally, we assume that the vessel follows the waterway. Accordingly, instead of measuring the positions within the waterways, we can also measure the distance to the starboard buoy d_s during a transition point TP, when crossing from one cell to the next. Figure 11 Visualization of Extracted Transition Points (TP) and $\boldsymbol{d_s}$ visualizes TP between the starboard s_1 and port buoy p_1 of a cell as well as d_s.



11 Visualization of Extracted Transition Points (TP) and d_s

Using d_s instead of the position based on latitude and longitude from the AIS data, simplifies the dataset since only one attribute instead of two corresponding attributes must be considered. Furthermore, the angle ß at the transition from one cell to the next holds information where the vessel is going next. This is shown in figure 12. where the vessel on the left wants to cross the waterway w_1 to enter another one w_2. In this case, the vessel has a different angle than the one on the right, following the waterway w_1.

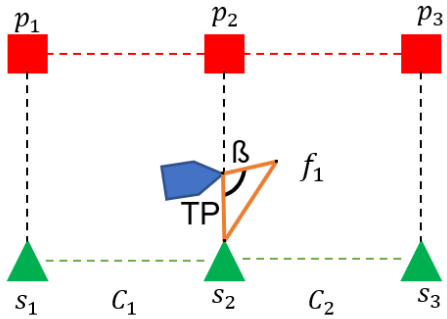


12 TP with Angle (ß)

To extract the information needed from the preprocessed AIS data, several steps are necessary. First the TP has to be calculated. For this purpose, the last position within the old cell and the first position within the new cell are selected. These two points form a line. The starboard and port buoy of the cell through which the positions pass also form a line. The transition point is then calculated by finding the point of intersection between these two lines.

In the second step, ß with which the vessel crosses the intersection point is calculated. Therefore, TP, the position of the starboard buoy s_2 as well as the first point after crossing the intersection point f_1 is used.

As it can be seen in 13, these three points form a triangle, from which ß can be calculated.



13 Calculation of ß

To calculate ß, the dot product formula can be used. Let TP, f_1 and s_2 be three given points and TPf_1 and TPs_2 be the two vectors formed by the points. The dot product of the vectors is then calculated as

$$TPf_1 \cdot TPs_2 = |TPf_1| * |TPs_2| * cos(theta) \quad \textbf{26}$$

where $|TPf_1|$ and $|TPs_2|$ are the magnitudes of the vectors $TPf_1$and TPs_2, respectively, and $theta$ is the angle between them. Solving for $theta$ leads to:

$$theta = arccos(\frac{\boldsymbol{TP}f_1 \cdot Ts_2}{|\boldsymbol{TP}f_1| * |TPs_2|}) \quad \textbf{27}$$

$Theta$ is provided as radian and is once again converted to degree:

$$ß_{\boldsymbol{degree}} = theta * (\frac{180}{\pi}) \quad \textbf{28}$$

Since the information of the SOG and COG attributes may also hold information about the future vessel navigation, these attributes should be used for the prediction. These characteristics are interpolated using the average of the last datapoint of the old cell and the first data point of the new cell. As interpolation value, the average of these two points is used. This basic process of creating TPs by AIS tracks and waterway data is illustrated as pseudo code in the appendix 3.

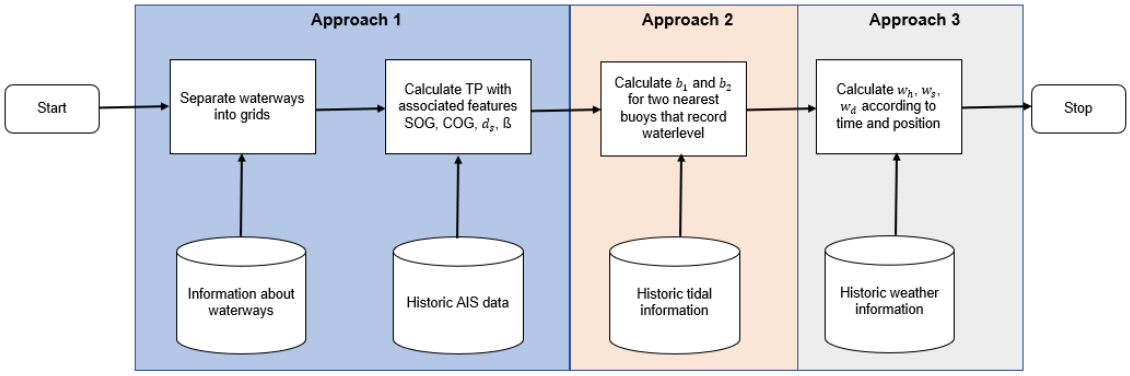
Another aim of this thesis is to use tide information to see if it can improve the accuracy of vessel track predictions. As mentioned in section 2.3, the depth of the waterway depends on the current tide, which then influences the track the vessel navigates. To check whether tide information has an influence on the prediction accuracy, it is combined with the existing dataset. Tide information is provided by the European Center for Medium-Range Weather Forecasts (ECMWF). These data consist of records of the water level, recorded at 10-minute intervals from buoys located at specific positions along the coast and in rivers. Depending on the position of the track, the measurements from the two nearest buoys are added to the AIS data. These two measuring stands are referred to as b_1 and b_2. Subsequently, the water levels at the buoys are determined time-dependently for each TP within the tracks. The previous water level and the following water level of a TP are determined within the 10-minute interval and an average water level is calculated. b_1 and b_2 are then added as features to the exiting dataset. The intention is that the model can recognize dependencies between the newly added features and the existing features during training, making the prediction of future TPs more accurate.

Beside the tide information, weather data should also be added to the dataset, with. wave height, wind speed and the wind direction being of particular interest. These

characteristics can be downloaded from the Era5 dataset mentioned in section 2.2, which consists of hourly estimations of these characteristics. The Era5 dataset consists of latitude-longitude grids with a 0,25° x 0,25° resolution [67].

The wave height w_h is available in meters. As explained in section 2.2, the wave height is an estimation based on the wave model WAM. The wind speed w_s is calculated based on the vector wind, which is the horizontal wind component in both the eastward and northward directions. It is measured through numerous instruments and provided in meters per second. The wind direction w_d is available in radians and also measured through various instruments. These characteristics are then added to the dataset, according to the time and position of the transition point.

Figure 14 summarizes the process of generating TPs within waterways and adding tide data and weather information.



14 Process of Generating TPs

In the first approach, a dataset will be generated using information from the waterways in combination with AIS data. This includes the attributes SOG, COG, d_s and ß, which represents TPs inside waterways. This data should then be augmented with tide information b_1 and b_2 as described earlier. In the third step, w_h, w_s, and w_d should then be added to the dataset so that the resulting dataset contains information about weather as well. This dataset should then be used to train data-driven models to predict future TPs. This prediction task can be considered as a regression task, where the dependencies of the features representing the transition point are to be learned and further predicted. As described in chapter 3, both LSTM models and transformer models can be used to solve these regression tasks, as they are able to learn dependencies between the given features and predict future TPs as accurately as possible. Thus, they fulfill the requirements that were set for the data-driven models.

5.2 Anomaly Detection

In this thesis, a data-driven model will be used to mark potentially anomalous transition points within waterways. To achieve this, each prediction for a transition point should be compared with the true transition point of the track. An anomaly in this context is a deviation from the prediction and the truth for one of the features SOG, COG, d_s and ß that exceeds a certain threshold. The individual thresholds are determined for each feature based on an analysis of the training data with which the model was trained. For this purpose, the mean μ and the population standard deviation σ are calculated for each feature SOG, COG, d_s and ß separately, where n is the total number of observations for all transition points and x is the value of the feature under consideration:

$$\mu = \frac{(x_1 + x_2 + \cdots + x_n)}{n} \quad \textbf{29}$$

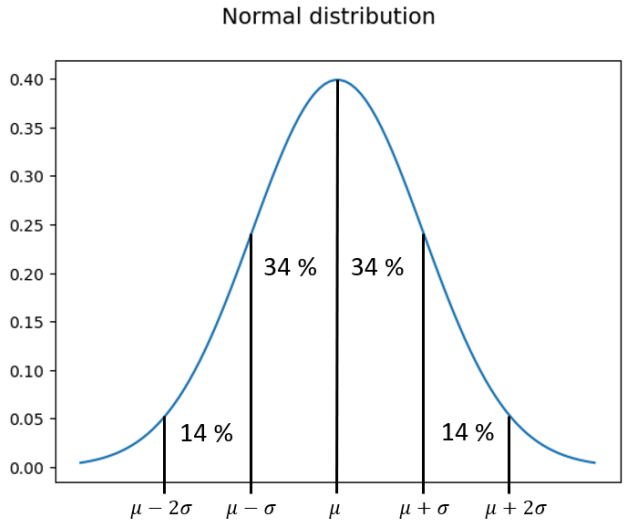
$$\sigma = \sqrt{\frac{\sum(X - \mu)^2}{n}} \quad \textbf{30}$$

From this, the normal distribution, which describes the probability distribution of each characteristic, can be derived,

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x - \bar{x}}{\sigma})^2} \quad \textbf{31}$$

In this normal distribution, the mean value represents the center of the distribution and the standard deviation represents the dispersion of the values around the mean value.

Based on this, it is now possible to determine the percentage of values present in certain areas, which is shown in figure 15. Here, the empirical rule can be used [68]. This statistical rule states that within one standard deviation ($\mu \pm \sigma$), 68% of the total observations fall within this range. 95% fall within the two standard deviations ($\mu \pm 2\sigma$) and 99.7% within the three standard deviations ($\mu \pm 3\sigma$).



15 Standard Deviation

Derived from this statistical rule, the threshold to detect anomalies can now be set depending on the requirement. The standard deviation indicates how much the data values deviate, on average, from the mean value of the dataset. In this concept, two standard deviations (2σ) is set as a threshold from which a deviation from the actual value should be marked as an anomaly. This value is taken because it is easy to determine and still reflects the characteristics of the distribution of the data. Setting the standard deviation as the threshold means that about 5% of the occurring values would also be recognized as anomalous. However, the approach presented here allows the threshold to be set variably. As Kumpulainen [69] pointed out, expert knowledge in the area of the application is required to set a meaningful threshold. It should be further emphasized that an anomaly in this thesis is defined only as a deviation from a certain threshold. This does not mean that a detected deviation represents a specific risk to vessel traffic within the predicted waterway.

6 Evaluation

In this chapter the developed concepts are evaluated. A dataset is generated based on the presented concept. This dataset is then used to train a Bi-LSTM and a transformer model, which can then be used to predict vessel tracks. The structure of the models is presented in section 6.2. The predictions are then compared and the results are evaluated. Furthermore, it will be evaluated whether anomalous vessel behavior can be detected with one of the models. For this, the previous presented concept is implemented and the results are discussed.

6.1 Data Preprocessing

In order to create a dataset that can be used to predict transition points within waterways, AIS data are needed. In this thesis, commercially available AIS data that were collected from terrestrial and satellite sources from 1 January 2020 to 30 April 2020 along the German Bight and the Elbe and Weser rivers were used as a basis. The region of interest is an area between 54.4°N, 10,7° E, 53.1° S and 5.8° W. As mentioned in section 5.1, the attributes considered from the AIS data were position, SOG, COG, MMSI and time. Figure 16 shows the first 200,000 positions from the January 1, 2020 AIS data in the area of interest.

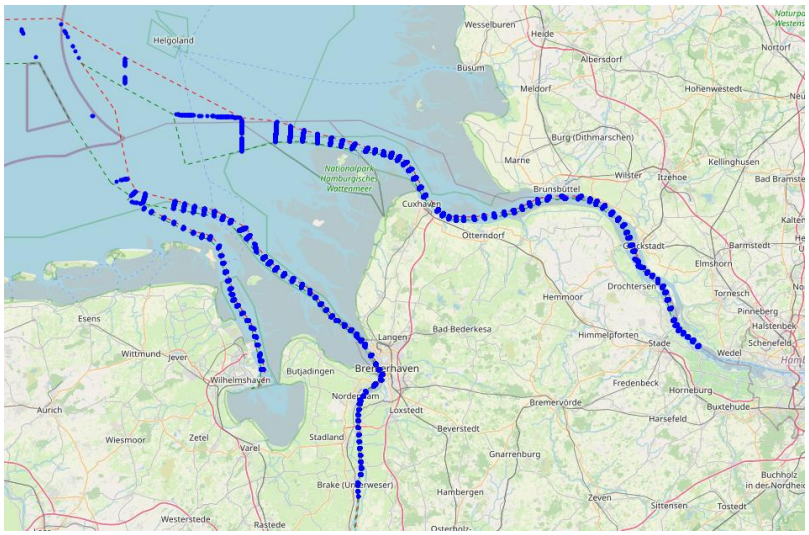
16 AIS Positions

As set out in section 2.1, AIS data may contain missing or corrupted values that can negatively affect the accuracy of subsequent analyses. Therefore, it is important to remove implausible values. Specifically, for the SOG attribute, values below 2 knots and above 30 knots were identified as outliers and removed from the dataset. Similarly, for

the COG attribute only values between 0 and 360 degrees were considered feasible and any values outside of this range were removed.

Next, these AIS data are combined with information about waterways. As can be seen in figure 1 in section 2.3, the waterways under consideration run north from the East Frisian Islands to Wilhelmshaven, along the Weser River to Bremen and along the Elbe River to Hamburg. According to the process described in section 5.1, the AIS data and the positions of the buoys within the waterways are used to create tracks of vessels, which in turn contain TPs along the waterways. These TPs further contain the associated features SOG, COG, d_s and ß.

The resulting dataset, as it can be seen in figure 17, now displays a list of transition points inside the waterways, where the blue dots display the feature d_s.



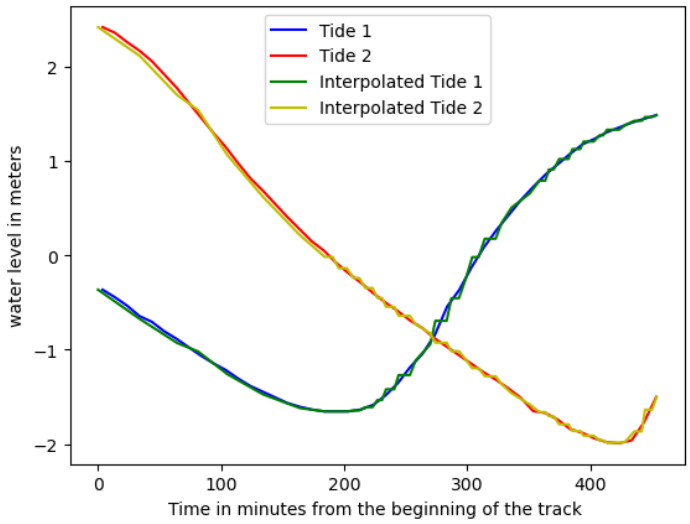
17 Extracted TPs

In the next step, tide information is added to the existing dataset. The tide information used is provided by the European Center for Medium-Range Weather Forecasts (ECMWF). These data consist of records of the water level, recorded in 10-minute intervals, from buoys located at specific positions along the coast and in rivers. Figure 18 displays the position of the used buoys in the area of interest.

18 Tide Buoys

Depending on the position of the track, the measurements from the two nearest buoys are added to the AIS data. This means that the tracks along the Elbe get assigned the measuring levels of the buoys St. Pauli and Cuxhaven, the tracks along the Weser get assigned the measuring levels of Bremen and the Alte Weser buoy. The tracks to or from Wilhelmshaven get assigned the measuring levels of the buoys Wilhelmshaven and Wangerooge. These two measuring stands are referred to as b_1 and b_2.

Subsequently, the water levels at the buoys are determined time-dependently for each transition point within the tracks. The previous water level and the following water level of a transition point are determined and an average value is calculated. Figure 19 shows the course of the depths of the two closest buoys and the interpolation during an example track.



19 Tide Interpolation

In the next step, wave height w_h, wind speed w_s and the wind direction w_d are added to the dataset. These characteristics can be downloaded from the Era5 dataset mentioned in section 2.2, which consist of hourly estimations of these characteristics. The Era5 dataset consist of latitude-longitude grids with 0,25° x 0,25° resolution [67]. The characteristics are then added according to the time and position of the AIS data point. Should one of the attributes not be available for the existing AIS position, data will be taken from the nearest grid that contains these attributes. This allows the weather attributes to be added to each AIS data point.

An important preprocessing step in machine learning is data scaling. During this process, the values of the preprocessed dataset are transformed to a specific range or distribution. The main objective of data scaling is to ensure that the features have similar scales and ranges, which allows the data-driven model to learn more effectively and ensures that all features contribute equally to the learning process. Without scaling, some features may dominate the learning process, leading to biased and inaccurate predictions. In this thesis, the data is scaled using min-max scaling. Thereby, the data is scaled between 0 and 1. The formula for min-max scaling is given as

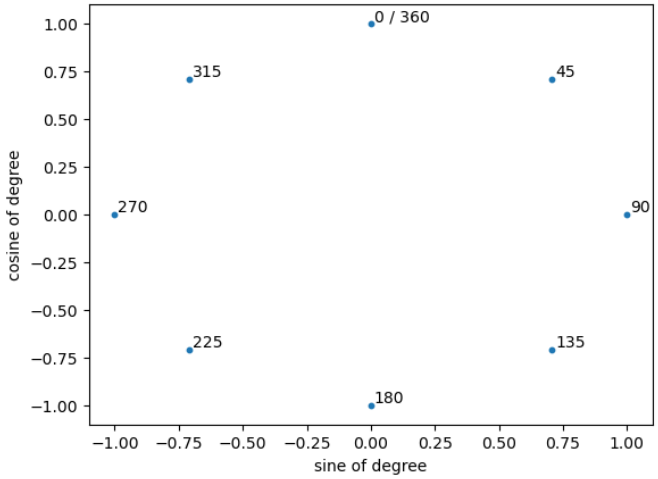
$$scaled_x = \frac{(x - min(x))}{(max(x) - min(x))} \quad \textbf{32}$$

The minimum value is subtracted from each data point x and the result is then divided by the difference between the maximum and minimum of all data points. This scaling is carried out for the attributes $SOG, d_s, b, b_2, w_h, w_s$.

Alternatively, to min-max scaling, other methods can also be used to scale data. Log scaling, for example, applies a logarithmic function to the data, which can help to reduce

the impact of extreme values and make the data more normal. Standard deviation can also be used. This method scales the data so that the mean of each feature is zero and the standard deviation is one. However, since the outliers of the AIS data as well as of the weather and tide data had already been reduced and the data features have a known range, min-max scaling was used in this thesis for the features $SOG, d_s, t_1, t_2, w_h, w_s$.

However, because the attributes $COG, ß$ and w_d are measured in or converted to degrees, scaling these values with a min-max scaling may not lead to more precise results because 0 and 360 point in the same direction but are maximally far away on the scale. Therefore, another approach was used. As it can be seen in figure 20, degree can also be displayed using sine and cosine.



20 Degree as Sine and Cosine

This representation was used to transform the attributes measured in degrees into sine and cosine using

$$a_s = sin(\frac{2\pi x}{360}) \quad \textbf{33}$$

and respectively

$$a_c = cos(\frac{2\pi x}{360}) \quad \textbf{34}$$

Although this means that the model has to predict two interdependent values instead of one, a more accurate prediction is expected from this representation.

In summary, by processing and merging the data as set out in the previous steps, a dataset summarized in table 7 is created:

Attribute	Description
SOG	Speed Over Ground in knots is interpolated with the last data point before the transition and the first data point after the transition and is. scaled with min-max scaler to normalize the values.
COG	Course Over Ground in degrees, like SOG, is interpolated with the last data point before the transition and the first data point after the transition and is then further divided into sine-COG and cosine-COG.
d_s	Euclidean distance between the transition point and the starboard buoy represents the proximity of the vessel to the starboard buoy and is scaled with min-max scaler to normalize the values.
ß	the angle at which the vessel crosses the starboard and port buoy is further divided into sine-ß and cosine-ß.
t_1, t_2	The water level in meters of the two nearest buoys along the waterway is modified with a min-max scaler.
w_h	Wave height in meters is scaled with min-max scaler
w_s	Wind speed in meters per second is scaled with min-max scaler.
w_d	Angle of the wind direction is further divided into sine-w_d and cosine-w_d.

7 Summary of Preprocessed Data

Overall, 11,167 tracks with at least 20 and up to 60 transition points are used. Thereof, 2,877 tracks (26%) navigate along the Weser, 7,949 tracks (71%) along the Elbe and the remaining 341 tracks (3%) navigate to or from Wilhelmshaven.

In this thesis, the models will be trained to predict the next transition point based on the last 10 transition points. In order to be able to use all transition points of the individual tracks for the training of the models, the tracks are divided using a sliding window approach in such a way that they always have a length of 10. Thus, there are a total of 205,380 tracks each with a length of 10 and the desired features. These tracks were then divided into a training set (80%) and test set (20%). In absolute numbers, this means that 164,304 tracks are used as training data and 41,076 as test data.

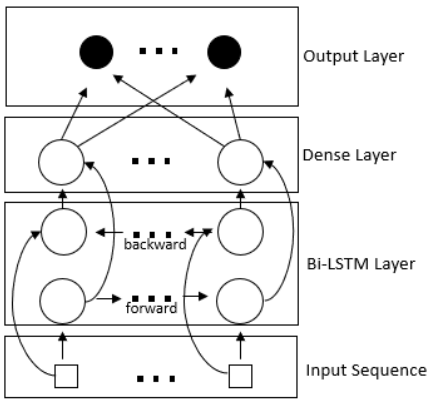
This data is then used with three different combinations of features. First, the AIS data combined with data extracted from the waterways should be used. This includes the attributes $SOG, COG, d_s, ß$. In the second step, tide data is added to the dataset to see if these additional data have an influence in the prediction accuracy b_1, b_2. Finally, the weather attributes w_h, w_s, w_d are also used. Using these three different combinations of features, will deliver insights into which combination of features achieves the best prediction result when used by data-driven models.

6.2 Models

In this thesis, the prediction of vessel tracks is tested with both a Bi-LSTM model and a transformer model. Furthermore, a linear prediction is used as a base model to better compare the obtained results.

The linear model is designed to take the feature it should predict as input and then perform a linear prediction on it. Accordingly, no other features that could affect the prediction are added to the model. Each sequence that the model gets as input is a 2-dimensional vector, where the first dimension displays the sequence length and the second dimension contains the feature that should be predicted. The input data is then flattened into a single dimension using the Flatten layer, resulting in a one-dimensional vector. This flattened input is then passed through a single dense layer with each output feature as neuron. Each neuron performs a linear transformation of the input data, which means that the output of each neuron is simply the weighted sum of its input, without any non-linearity.

The architecture of the Bi-LSTM model is displayed in figure 21. The model contains one Bi-LSTM layer, a dense layer and finally an output layer and is based on the structure explained in section 3.3.

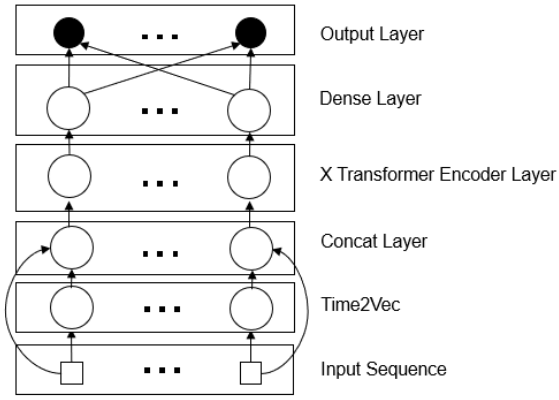


21 Bi-LSTM Architecture

The Bi-LSTM layer contains 128 units and is the core of the model. It processes the input data sequentially, considering the sequence of input data in addition to the individual features at each time step. As mentioned in section 3.3, the Bi-LSTM contains two separate LSTM layers, one processing the input sequence in a forward direction and the other in a backward direction. This allows the model to capture temporal dependencies in both directions and can improve the accuracy of predictions. Further, a dropout layer is added to prevent overfitting during training. The dropout rate was set at 0,2, meaning that 20% of neurons randomly dropped out during training. This layer is then followed by a dense layer with 64 units which processes the information from the Bi-LSTM layer. Finally, a dense layer with the expected features was used that outputs a tensor and applies a linear transformation to the output of the previous layer.

The number of units as well as the hyperparameters used in the model were found out with the Keras Tuner [70]. Keras Tuner is a hyperparameter tuning library for TensorFlow's Keras API. It allows to efficiently search for the best hyperparameters for deep learning models. Keras Tuner uses various search algorithms and strategies to find the optimal hyperparameters for a given model architecture and training dataset. For this purpose, a list of parameters can be passed which are then tried out. The models with which the best results were achieved can then be trained further.

The structure of the transformer model is shown in figure 22.



22 Transformer Architecture

For the transformer model, the input sequence is first fed into a Time2Vec layer, which works as described in section 3.5. However, since time is not considered, the mean value from all values of the respective features is used. From this mean value, two features are then calculated that represent the position within the sequence. These additional two features are then concatenated to the input sequence. The output of this layer is then fed to three consecutive transformer encoder layers. The transformer encoder layer is

based on the transformer model introduced in section 3.5, where the input is fed to a Multi-Head Attention layer and then followed by a Normalization and Dropout layer. The dropout rate was set to 0,1. Next, a feed forward layer is implemented as a convolutional layer. The purpose of this layer is to provide each encoder block with the ability to learn and model complex interactions between the input sequence elements. The output is then again normalized. After the encoder layers, a global average pooling layer follows. This layer reduces the tensor to a single scalar value by taking the average of all its values across a particular axis. This layer reduces the dimensionality of the output while preserving important spatial information. A dropout layer is then used with the dropout rate was set to 0,2. Finally, like in the Bi-LSTM model, a Dense layer with 64 units followed by a final dense layer with the required number of features is added to the model.

6.3 Implementation

Various frameworks and publicly available libraries were used during data preprocessing, creating and training the models and in the evaluating process. Python was used as the main programming language. The libraries and frameworks mainly used are introduced in the upcoming paragraphs.

TensorFlow + Keras

The data-driven models in this thesis are implemented using TensorFlow. TensorFlow is an open-source framework developed by Google Brain that provides tools for building and training machine learning models. TensorFlow can be used on different platforms, including mobile devices and the web. It also enables the use of Keras, which is a high-level neural network application programming interface (API). Keras allows to easily build, train and deploy neural networks, using an API that hides the low-level details of the underlying deep learning framework. It supports a wide range of neural network types, including convolutional neural networks (CNNs) and recurrent neural networks (RNNs) among others. Keras also supports a wide range of loss functions, activation functions and optimizers which are used to train the neural networks. [71] [72]

Pandas

Pandas is an open-source library for data manipulation and analysis in Python. It provides data structures for efficiently storing and manipulating large datasets, as well as tools for reading and writing data in various file formats. It further provides a rich set of functions and methods for performing data analysis tasks, such as filtering, grouping

and aggregation. It also supports data visualization through integration with other libraries such as Matplotlib. [73]

GeoPandas

GeoPandas is a Python library that builds on top of pandas and adds support for working with geospatial data. It provides a convenient and efficient way to manipulate and visualize geospatial data in Python. Geospatial data includes data that is associated with geographic locations on the Earth's surface, such as latitude and longitude coordinates, polygons, points, and lines on a map. GeoPandas provides tools for reading, writing, and manipulating geospatial data in a tabular format, similar to the way that pandas works with numerical and textual data. [74] In this thesis, GeoPandas is used to process data that includes geographical positions like AIS data or the positions of the buoys inside the waterways.

Matplotlib

Matplotlib is a data visualization library for Python. It provides a wide range of tools for creating static, animated and interactive visualizations in Python. Matplotlib allows to create high-quality 2D and 3D graphs, plots, histograms, scatterplots and many other types of visualizations. It is open-source and available under the BSD license, which means it is free to use, distribute and modify. [75] The figures presented in this thesis are mainly made using Matplotlib.

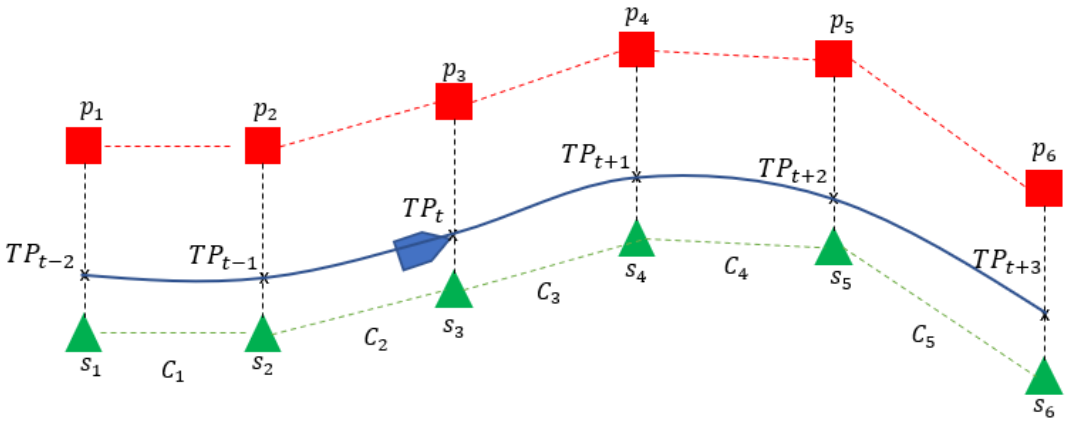
NumPy

NumPy is a Python library used for numerical computing. It stands for "Numerical Python" and is designed to provide fast, efficient operations on large multi-dimensional arrays and matrices of numerical data. NumPy is a fundamental library for scientific computing in Python and is used extensively in fields such as data science, machine learning, physics, engineering and finance. [76]

6.4 Vessel Track Prediction

In this section, three different feature combinations are trained and evaluated using a Bi-LSTM and a transformer model. These models are then also compared to the linear prediction result for each feature. The linear prediction will serve as a base model to better classify the results. The processing of the data to combine the different features is explained in section 6.1, while the structure of the two data-driven models on which the data was trained is explained in section 6.2.

As mentioned in chapter 5.1, the prediction of vessel tracks can be categorized as a regression task. In this thesis, a sequence of TPs is given from which the subsequent TPs are to be predicted. Thus, this task can be placed in a temporal context. The time sequence of a vessel track is shown in figure 23, where TP_t represents the transition point at timestep t.



23 Vessel Track Sequence

In this thesis, the subsequent five TPs should be predicted. This prediction process can be displayed with equation 35, 36, whereby TP also includes the other features of the dataset.

$$TP_{t+1} = f(TP_t, TP_{t-1}, \dots, TP_{t-9}) \quad \textbf{35}$$

$$TP_{t+2} = f(TP_{t+1}, TP_t, \dots, TP_{t-8}) \quad \textbf{36}$$

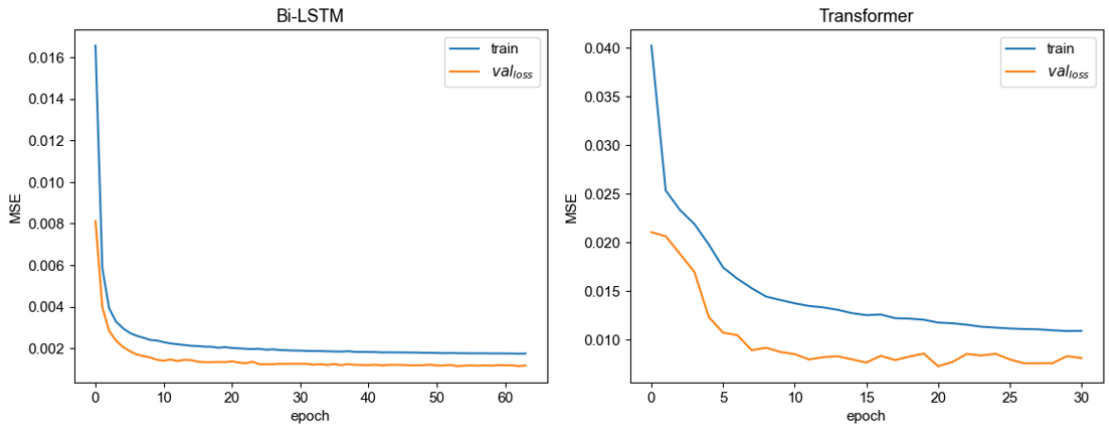
To predict the next transition point TP_{t+1}, the ten most recent transition points are used $(TP_t, TP_{t-1}, \dots, TP_{t-9})$. This predicted transition point is then incorporated with the previous nine transition points to make further predictions for the following transition point TP_{t+2}. This iterative process is repeated to predict the next five transition points.

Each model is trained for a maximum of 100 epochs. However, if the result do not improve during the last 10 epochs, the training is stopped and the best weights are restored. Additionally, each model is trained using the Adam optimizer as it is well suited for these kinds of predictions and is more efficient than, for example, the SDG optimizer. The MSE is used to measure loss, obtaining more accurate results compared to the MAE. During training, a validation split of 0,2 is used, leading to 20% of the training-data being used as a validation set. This set is then used to evaluate the performance of the model during training, which is measured as validation loss val_{loss}.

6.4.1 Results

In the first approach, the Bi-LSTM and the transformer model were trained on a dataset containing the attributes SOG, COG, d_s and ß. As mentioned in section 6.1, COG and ß were further divided into sine and cosine so that this dataset contains six features. Overall, the training data had 164.304 different tracks with a sequence length of 10. The test data had 41.076 example tracks with the same sequence-length and features as the training data.

When comparing the two different models, it is noticeable that the transformer model with 24.286 trainable parameters is by far the less complex model, while the Bi-LSTM model has 155.078 trainable parameters. This complexity is reflected in the training of the models. Figure 24 shows the training and val_{loss} of the two models during the training per epoch.



24 Approach 1: Training and Validation Error

As can be seen in both models, the training error decreases significantly at the beginning and only slightly thereafter. It is noticeable that the transformer model on the right already converges after 30 epochs, while the Bi-LSTM model finally converges after 63 epochs. When considering the level at which the models converge, the complexity of the Bi-LSTM leads to a better result compared to the transformer model. As table 8 illustrates, at 0,001751 the Bi-LSTM model reaches a lower MSE than the transformer model at 0,010887. The same applies to the val_{loss}. It is below the trainings loss but does not show signs of over- or underfitting.

Model	MSE	val_{loss}
Bi-LSTM	0.001751	0.001176
Transformer	0.010887	0.00808

8 Approach 1: MSE & Validation Loss

Considering the average prediction error among the next five transition points with respect to the features SOG, COG, D_s and ß, as shown in table 9, the results measured with the MSE give a good indication of the achieved average result per feature. In addition to the Bi-LSTM and the transformer model, a linear prediction of each feature is used a baseline to which the data-driven models can be compared.

Model	D_s (meters)	SOG (knots)	COG (degree)	ß (degree)
Linear	178.09	0.74	9.97	18.93
Bi-LSTM	106.42	0.68	2.07	1.85
Transformer	182.75	2.73	5.24	7.09

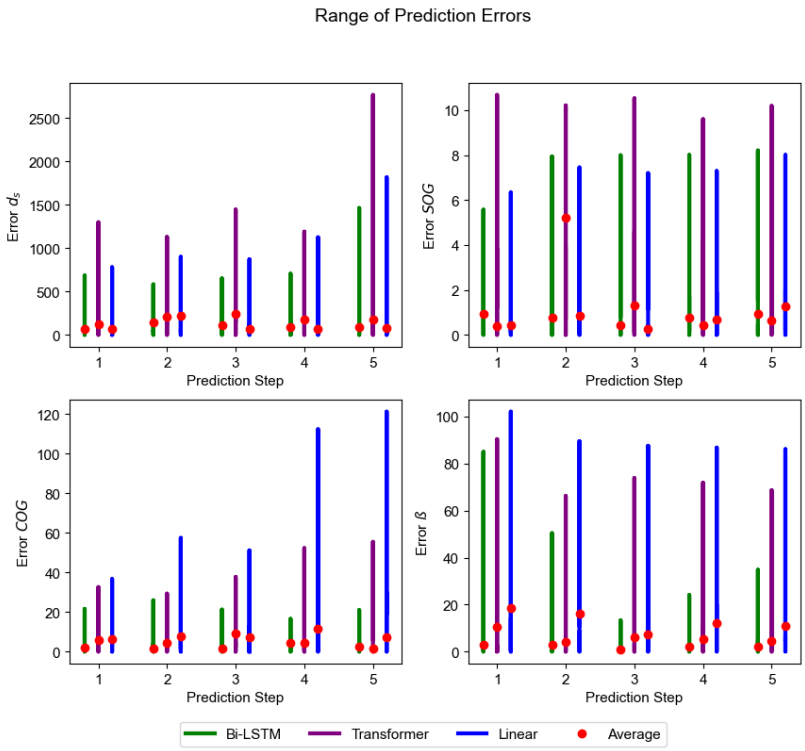
9 Approach 1: Results per Feature

The best result in terms of all features is measured with the Bi-LSTM. The transformer model, on the other hand, makes more precise predictions than the linear prediction for COG and ß, but less precise predictions for d_s and SOG.

The average error for the Bi-LSTM for d_s is 106,42 meters, which is more than 70 meters more accurate than the linear prediction and the transformer model's prediction. However, the linear prediction and the transformer model achieve very comparable results of 178,09 meters and 182,75 meters, respectively. For the average error of SOG, the linear prediction (0,74) and the Bi-LSTM model (0,68) are very close. However, when predicting COG and ß, there are three clearly different results. The lowest average error is achieved by the Bi-LSTM model with an error of 2,07 degrees for COG and 1,85 degrees for SOG. This is followed by the transformer model with an average error of 5,24 degrees for COG and 7,09 degrees for ß. The linear average errors are significantly higher with an average error of 9,97degrees for COG and 18,93 degrees for ß.

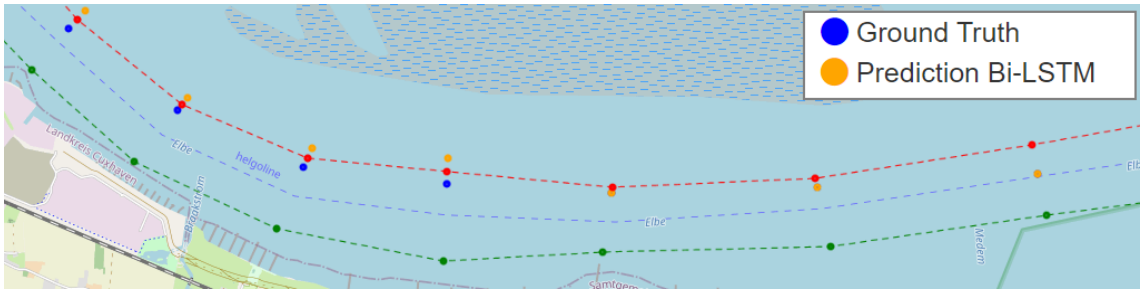
The following figure 25 shows the range of errors for each prediction step and for each feature as well as the average of the prediction errors from the Bi-LSTM, the transformer model and the linear prediction. As can be seen, the average error of all features does not consistently increase when predicting the later transition points. This is of interest since the predicted errors are included in the later prediction. However, the outliers in the prediction of COG and ß in the Bi-LSTM become larger in later predictions, as well as in the prediction of d_s with the transformer model.

For the features d_s and SOG, the transformer model predicts the largest error range, with an error for d_s of up to 2.700 meters relative to ground truth and up to 11 knots for SOG. It is noticeable that the average error for SOG for the transformer model increases significantly in the second step and then decreases again. When considering COG and ß, the linear prediction predicts large outliers of up to 120 degrees for COG and 100 degrees for ß.



25 Approach 1: Range of Prediction Errors

Looking at individual tracks, it is noticeable that all predictions can produce erroneous results. As shown in figure 26, the Bi-LSTM model predicts values outside the waterway.



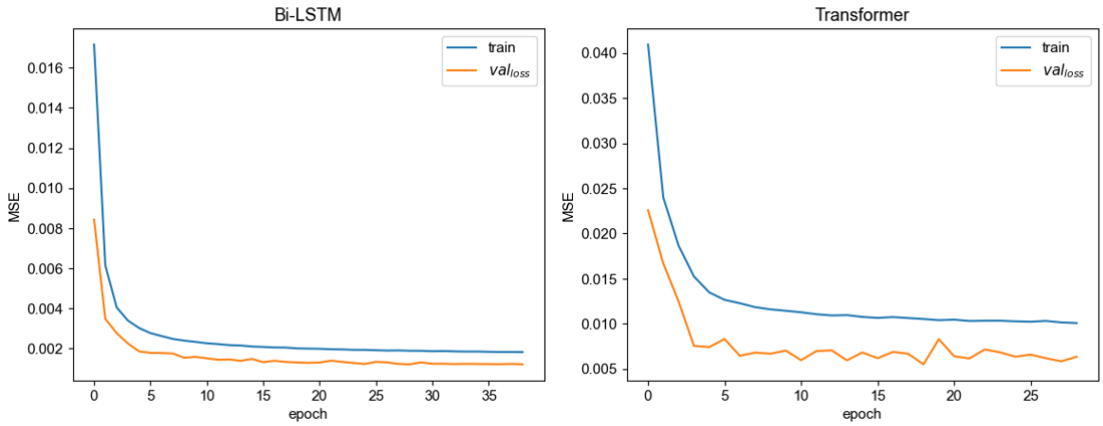
26 Erroneous Track Predictions

This phenomenon occurs in the Bi-LSTM model in 0,7% of all test predictions and in the predictions done with the transformer model in 0,4% of all test predictions. The linear prediction only produces this error in 0,4% of all test predictions.

For the next predictions, the tide level of the two nearest buoys b_1 and b_2 were added to the dataset as mentioned in section 5.1. Accordingly, the feature set of each model increases by two to eight features. Due to this increase, the trainable parameters of the Bi-LSTM model increase by 2.178, to a total of 157.256, while 8.766 additional parameters were added to the transformer model, which thus contains a total of 33.052 parameters.

During training, the MSE was adjusted so that only the previous attributes SOG, COG, D_s and ß were included in the error and not the added attributes b_1 and b_2. Although the errors of the tide predictions are higher when excluding them from the loss function and the tidal predictions are included in the further predictions, this approach obtained more accurate results compared to models trained on all features.

The progression of the training curve, shown in figure 27, is comparable to that in the first approach without tide information. The validation error of the transformer model has a higher volatility than that of the Bi-LSTM model and the transformer model already converges after 28 epochs while the Bi-LSTM model takes 38 epochs.



27 Approach 2: Training and Validation Error

When looking at the results for the MSE and the val_{loss} in table 10, the values change only minimally compared to the training without tide information. Thus, the Bi-LSTM model remains the one with the lower MSE.

Model	MSE	val_{loss}
Bi-LSTM	0.001815	0.001208
Transformer	0.010076	0.006336

10 Approach 2: MSE & Validation Loss

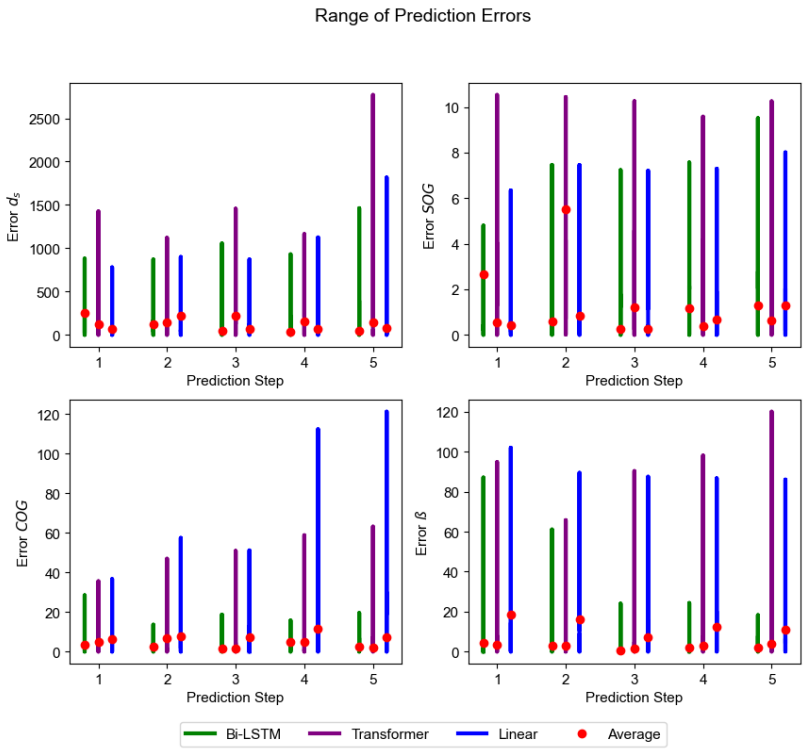
Although the MSE has changed only slightly, the prediction errors of the individual features of the Bi-LSTM are significantly worse than without tide information. This can be seen in table 11.

Model	D_s (meters)	SOG (knots)	COG (degree)	ß (degree)
Linear	178.09	0.74	9.97	18.93
Bi-LSTM	197.46	1.02	2.16	2.03
Transformer	171.21	2.75	5.06	5.95

11 Approach 2: Results per Feature

The transformer model performs slightly better for d_s than the linear prediction with an error of 171,21 meters. The Bi-LSTM model got worse and now predicts an average error of 197,46 meters. The lowest average error for SOG is now predicted by the linear prediction, then followed by the Bi-LSTM model. For COG and ß, the Bi-LSTM model still reaches the lowest average error, followed by the transformer model.

As for the predictions outside the waterway, the transformer model predicts for 1.3% of all tracks, negative distances to the starboard buoy. For the Bi-LSTM model, however, the erroneous predictions increase to 37% of the total test predictions.



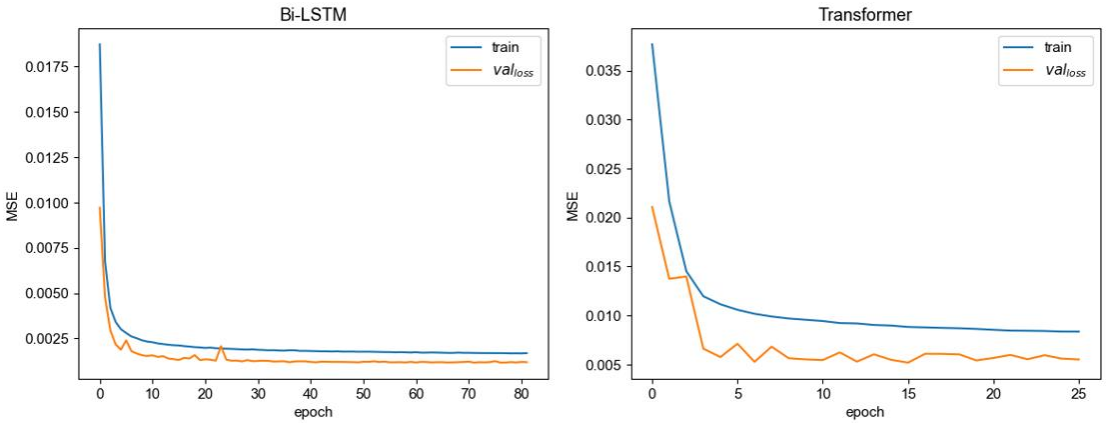
28 Approach 2: Range of Prediction Errors

When looking at the range of the prediction errors, displayed in figure 28, it is noticeable in the Bi-LSTM model that when predicting d_s, the range is increased compared to the prediction without tide information. With SOG, the average error especially in the first prediction step is higher than in the subsequent steps. With the transformer model, there is little change in the range of prediction of d_s and SOG. For ß, however, the ranges increase up to 30 degrees in the later predictions. This is also associated with the poorer average predictions.

For the last predictions, w_h, $w_s$and w_d, which represents the weather characteristics, were then added to the dataset as mentioned in section 5.1. Adding these additional

features increased the feature size to 12, because w_d was preprocessed like COG and ß, which means that it was divided into cosine and sine. Due to this change, the parameters of every model again increased. The Bi-LSTM now had 161.612 trainable parameters and the transformer model 53.784 trainable parameters.

The progression of the training and validation errors for the linear and Bi-LSTM is again very similar to the progression of the previous trainings, as it can be seen in figure 29.



29 Approach 3: Training and Validation Error

When looking at the results of the training, as displayed in table 12, it is noticeable that the MSE of the transformer model reaches with 0,001688 nearly the same results as during the training with the tide data. The Bi-LSTM model, on the other hand, achieves with 0,008354 a lower MSE than in the previous trainings.

Model	MSE	val_{loss}
Bi-LSTM	0.001688	0.001188
Transformer	0.008354	0.005513

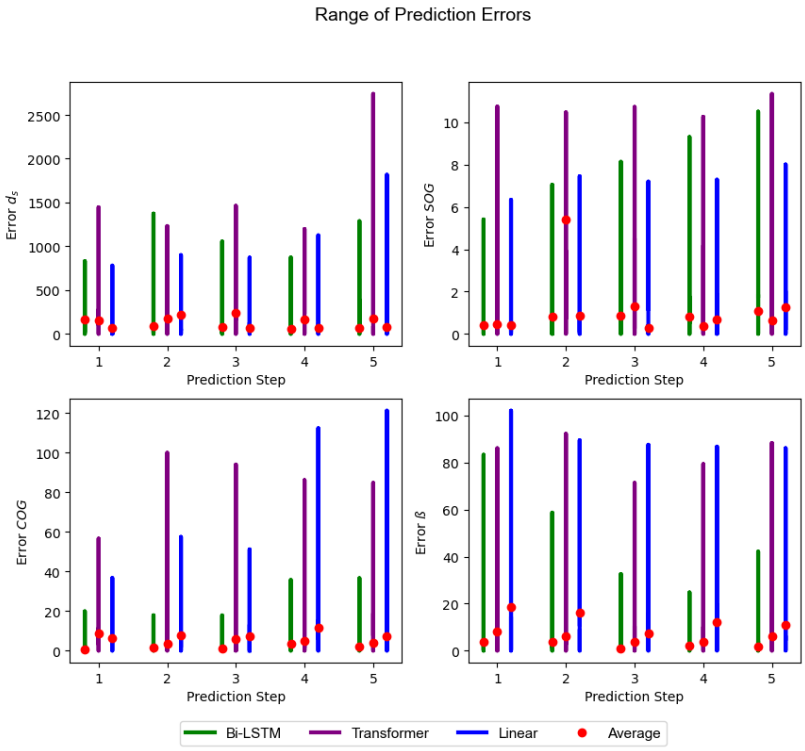
12 Approach 3: MSE & Validation Loss

Looking at the average errors of the prediction of the following five transition points, the following picture emerges (Table 13).

Model	D_s (meters)	SOG (knots)	COG (degree)	ß (degree)
Linear	178.09	0.74	9.97	18.93
Bi-LSTM	153.97	1.25	2.11	2.18
Transformer	185.67	2.79	7.4	7.64

13 Approach 3: Results per Feature

As the results show, the lowest prediction error for d_s is now received by the Bi-LSTM model. For SOG, the linear prediction reaches the lowest average error. For the features COG and ß, the three result categories can still be seen. The Bi-LSTM model achieves the lowest average error, followed by the transformer model and the linear prediction.



30 Approach 3: Range of Prediction Errors

Looking at the prediction error range in figure 30 for SOG, the average for the Bi-LSTM is higher in the first prediction step than in the subsequent ones. The error range for ß predicted by the transformer model is lower compared to the approach with tide data.

The average error and the total error range are shown again in detail in appendix 4 and 5 separately for the different approaches according to the Bi-LSTM and transformer model. While considering d_s for the Bi-LSTM, it is noticeable that the error range

increases for the second prediction step with the approach that includes tide data and weather information. However, despite the large range, the average error is lower as in the other approaches. This phenomenon can also be seen with COG during the fourth and fifths prediction step. When looking at SOG, there is a clear deterioration in the first two prediction steps with the addition of tide data. When looking at the transformer model, an increase of the error range with the addition of tide data and weather information for COG occurs from the fourth prediction step onwards. This is also accompanied by a decrease of the average error. Like the Bi-LSTM, the average error increases for SOG during the first prediction step, when adding tide data.

The following table 14 summarizes the collected results of each approach for each feature. As the results show, the Bi-LSTM in the first approach achieves the lowest deviation from the ground truth for every feature.

	Model	D_s **(meters)**	SOG **(knots)**	COG **(degree)**	ß **(degree)**
	Linear	178.09	0.74	9.97	18.93
Approach 1 $d\_s, COG, SOG,$ ß	Bi-LSTM	106.42	0.68	2.07	1.85
	Transformer	182.75	2.73	5.24	7.09
Approach 2 b_1, b_2	Bi-LSTM	197.46	1.02	2.16	2.03
	Transformer	171.21	2.75	5.06	5.95
Approach 3 w_h, w_s, w_d	Bi-LSTM	153.97	1.25	2.11	2.18
	Transformer	185.67	2.79	7.4	7.64

14 Overall Results per Feature

When considering d_s, the prediction of the Bi-LSTM model in the first approach, in which only the features SOG, COG, d_s and ß were used, achieves an average prediction error of 106,42 meters. This result is about 65 meters more accurate than the best result achieved with a transformer model, which is achieved in approach 2 where water levels from two nearest buoys b_1 and b_2 were added. The linear model achieves an average prediction error of 178,09 meters, which is nearly 70 meters less accurate than the result achieved with the best Bi-LSTM model.

When considering SOG, the best result is also obtained from the Bi-LSTM in the first approach by adding the tide data. On average, the average prediction error there is 0,68 knots. It is noticeable that the result does not differ much from that of the linear prediction, which reaches a deviation of 0,74 knots. The transformer model, on the other hand, delivers a higher prediction error at 2,72 knots.

For COG and the ß, the Bi-LSTM achieves 2,07 and 1,85 degrees respectively in the first approach, which is by far the lowest prediction error compared to the other predictions. The Bi-LSTM model is followed by the transformer model, which achieves an average prediction error of 5,06 degrees for COG and 5,95 degrees for ß. This is achieved in the second approach. The largest deviation with 9,97 degrees for COG and 18,93 degrees for ß is obtained with the linear prediction.

It is noticeable that the prediction results of the Bi-LSTM model become worse when more data features were added. However, the prediction results of the transformer model became better when tidal data were added. However, when adding weather data, the results also got worse.

In table 15, the predictions obtained with the Bi-LSTM model, which obtained the best result in comparison, are divided according to the waterways along the Elbe and Weser rivers and Wilhelmshaven.

	D_s (meters)	SOG (knots)	COG (degree)	ß (degree)
Elbe	100.57	0.65	2.17	1.87
Weser	115.15	0.71	1.90	1.76
Wilhelmshaven	105.15	0.82	2.22	2.64

15 Results per Feature per Track

As the results show, the lowest average prediction error for d_s is achieved along the Elbe river with 100,57 meters, followed by the predictions along Wilhelmshaven with 105,15 meters and the Weser river with 115,15 meters. For SOG, the lowest error is again obtained along the Elbe. However, the difference between the best and the worst result applicable to Wilhelmshaven is only less than 0,2 knots

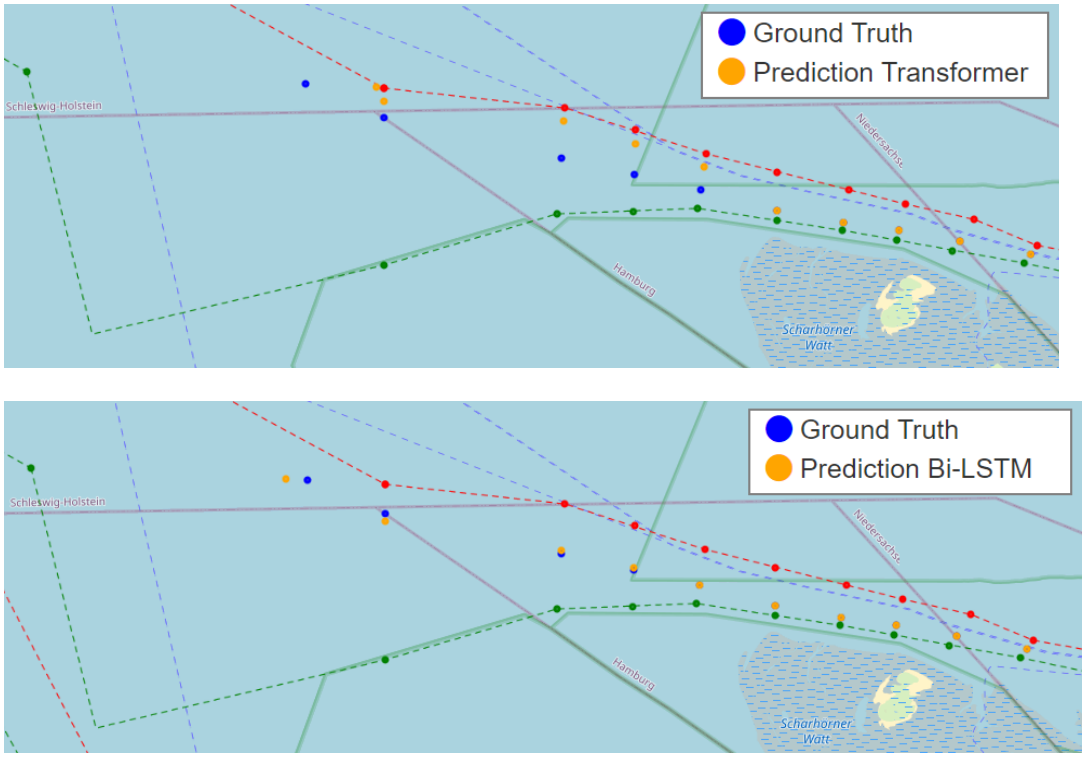
For the other features COG and ß, the lowest error is received along the Weser River. The prediction results for these two features are, with less than 1-degree difference, also very close to each other.

6.4.2 Discussion

The discussion of the results can be divided into four parts. In the first part, the history of the training and validation errors for the three approaches is discussed. Second, the outliers in the prediction as well as erroneous tracks are discussed, followed by a short discussion about the prediction errors per waterway. Finally, the results are evaluated against the requirements established at the beginning of this thesis.

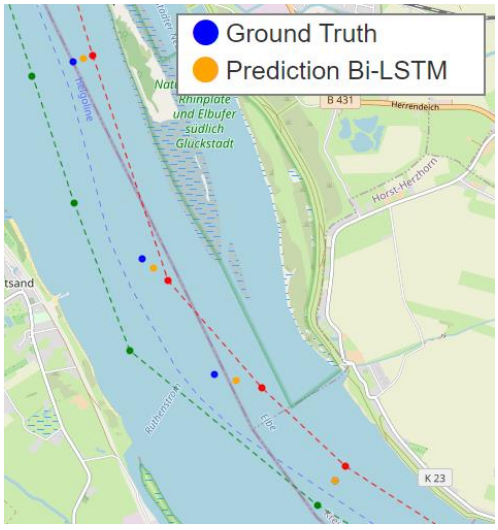
When looking at the training and validation errors in figures 24, 27 and 29, the transformer model converges much faster than the Bi-LSTM model. The faster convergence is related to the complexity of the models since, as mentioned before, the Bi-LSTM has significantly more trainable parameters and thus takes longer to find an optimal solution. Considering the validation error of the transformer model in figure 27, the volatile performance may somewhat indicate an overfitting of the model. However, as the result does not deteriorate in the long run, so that the model should not be classed as overfitted. Attempting to make the transformer model more complex resulted in overfitting. Accordingly, a model is selected even though it has much fewer trainable parameters than the Bi-LSTM model. Accordingly, to make the transformer model more complex, a different architecture needs to be chosen. It is also noticeable that the MSE changes only slightly when tide data and weather information are added, whereas the specific predictions of the features do. Thus, it can be concluded that the MSE does not directly reflect the predictions of the individual features.

When considering the range of prediction errors, as displayed in figure 25, 28 and 30 for d_s, the transformer model predicts a wider range of errors than the Bi-LSTM model. These outliers in the transformer model's prediction of the d_s occur for tracks where the waterway becomes significantly wider. This can be seen in figure 31 where the d_s of the 5th prediction is 2.766 meters away from the actual position. In this case, the Bi-LSTM makes a much more accurate prediction, although it also predicts to an error of 263 meters. In the example at hand, the vessel was sailing far to the left in the waterway, which is unusual, since it was assumed that vessels sail as far to the right as possible. This deviation from the assumption of the vessel`s behavior may also contribute to the large difference between the predicted distance and the actual distance. Accordingly, the assumption needs to be reconsidered, as it may not hold for all vessel types considered in the data.



31 Outliers d_s

The outliers in the ß's and in the $COG´$s prediction error occur mainly for one location in the waterway along the Elbe river, which is displayed in figure 32. The waterway runs along Glückstadt and the buoys are arranged in such a way that it seems as if the ships do not pass through the waterway from the north or south, but from the west or east. Since the arrangement of the buoys in the course of the waterway only give the impression at this intersection, the models can not represent this transition well. Had the model actually predicted this outlier, this would have indicated that the model had overfitted and made predictions too close to the training data.



32 Outliers COG & ß

In the case of the COG, it is particularly noticeable in the first approach that the linear predictions show significantly greater deviations in the later predictions than the other two models. This is also accompanied by a worse than average prediction accuracy. This illustrates that a linear prediction does not provide good results for these features and that the other two models can predict the values significantly better.

As mentioned in the previous chapter, all predictions can produce erroneous results. This phenomenon occurs mainly in curve passages where the distance to the previous transition points becomes significantly smaller. With the introduction of tide information in the Bi-LSTM model 36% of all tracks were predicted erroneously, which indicates that with the addition of the features, the model becomes distracted in the prediction and can no longer represent the dependencies between the features as well as without tide information. This can also be seen from the fact that the average error in the prediction of d_s increases by almost 90 meters. However, the added features seem to help the transformer model, because all predicted features, except SOG, got more precise.

The comparison of the results of the different models can be further evaluated with the aid of figure 4 in the appendix. As shown in the figure, the range of d_s for the Bi-LSTM model increases with the addition of tide and weather data from the first prediction step onwards. The average error, however, is somewhat smaller from the second prediction step onwards compared to the experiment without tide and weather information. This suggests that the model has adjusted more closely to the training data, making the average predictions more accurate. The outliers from the data, on the other hand, can no longer be represented as well, which explains the higher errors. As a result, deviations from the norm lead to a larger error. This is also the case for COG during the fourth and fifth prediction step. For the other features, the error range do not change so much with the addition of the other data. For the transformer model, the same phenomenon is noticeable for COG in the fourth and fifth prediction step, as well as in the second prediction step for d_s as it can be seen in appendix 5.

All in all, the results displayed in table 14 show that the addition of tide information as well as weather characteristics does not automatically lead to an improvement of the predictions. One possible reason for that is that the new features may not be relevant or informative to the prediction task and therefore may introduce unwanted noise or bias to the model. Overall, it should also be noted that the Bi-LSTM predicts the lowest average error. On the one hand, this can be explained by the fact that it was also the most complex model and thus had more capacity to predict the dependencies. However, even the less complex transformer model makes reasonable predictions for the number of trainable parameters it possesses. In particular, the strength of a transformer model is

its ability to identify long-term dependencies [37]. In the presented approach, however, only the first following transition point was predicted. Accordingly, the models did not have to learn large dependencies. For this kind of prediction, LSTM models are also very well suited, as can be seen in the results. In summary, both data-driven models show better results than linear prediction, which justifies the use of these models for this prediction task. It is also worth noting that the average prediction error for the transformer model decreases when tide data are added, while the average prediction error of the Bi-LSTM model increases. That results in general can get worse with the addition of features has already been described by Kuhn et al. [77] and also by John et al. [78]. This is also the case in this thesis, especially for the weather data and the tide data for the Bi-LSTM model. This is the case even though tide and weather data have a real impact on the navigation of vessels. However, the data-driven models could not use this additional information to their advantage in order to make more accurate predictions.

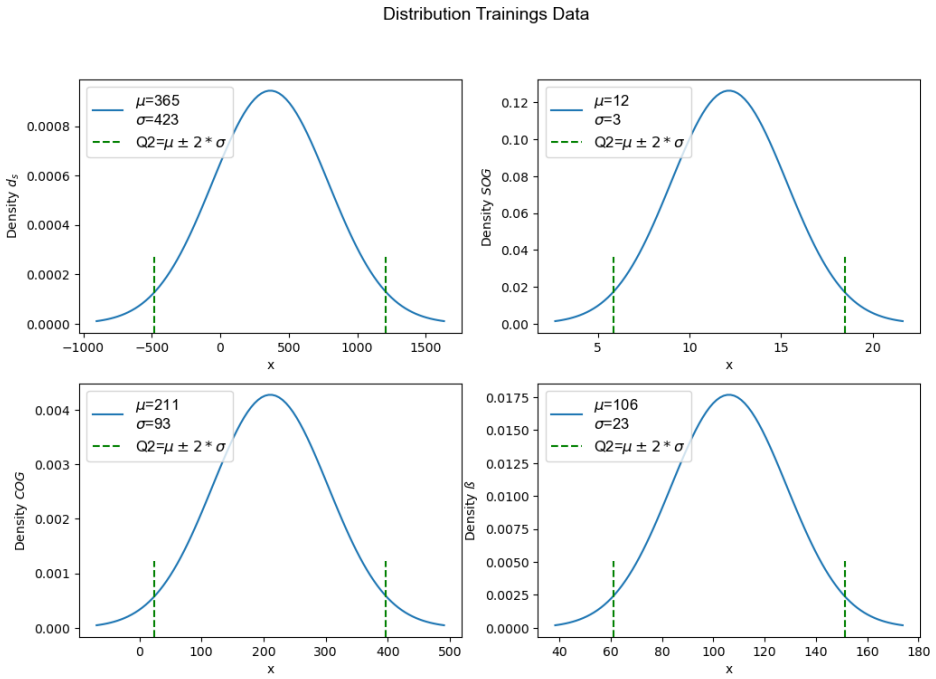
When the average error is split among the waterways considered, it is observed that only minor differences in the prediction results exist between the individual waterways. That the predictions to and from Wilhelmshaven is slightly worse than for the Elbe and Weser rivers, can be explained by the fact that the model was trained on only 3% of the total tracks in this waterway. Therefore, this waterway was underrepresented for the model. However, since the overall results of the predictions for the different waterways are very close, it can be concluded that the presented concept for predicting ship tracks in waterways can be used for more than one waterway.

As the results show, both the concept and the data-driven models meet the requirements specified for them at the beginning of this thesis. For the concept, the requirements included the combination of historical vessel data, weather information, tide data and sea chart information. With this data, data-driven models were to be trained to accurately predict further vessel tracks. In the concept presented, AIS data along the German coast were selected as historical vessel data, which were then combined with weather data and data on waterways extracted from sea charts. Vessel tracks were therefore defined as subsequent TPs points inside waterways. The data-driven models were required to make predictions especially in the coastal areas. Further, they were to be trained with the data generated from the concept and to be able to predict vessel tracks accurately. A Bi-LSTM and a transformer model were selected as data-driven models and trained on the data generated from the concept. The results were then compared to those from a linear prediction. Results showed that both models make more accurate predictions than the linear model and are therefore better suited for the prediction of ship tracks. It

was also discovered that the addition of tide and weather information did not lead to an overall improvement of the forecast results.

6.5 Anomaly Detection

To evaluate the concept of anomaly detection presented in chapter 5.2, the data on which the model was trained is analyzed. Therefore, the mean μ and the standard deviations σ are calculated for the features d_s, SOG, COG and ß. Accordingly, using μ and σ, the normal distribution is calculated, as displayed in figure 33, where x is the value of the feature.



33 Normal Distribution Training Data

As mentioned in the concept section, a track should be marked as anomalous if the difference between the predicted feature and the true feature differs by more than two times the respective sigma. Specifically, a track is flagged as an anomaly if d_s deviates by more than 846 meters, SOG by more than 6 knots, COG by more than 186 degrees, and ß by more than 78 degrees.

In order to test the concept of anomaly detection, the model that provides the most accurate predictions will be used. As pointed out in section 6.4, this has turned out to be the Bi-LSTM model, which was trained with the parameters SOG, COG, d_s and ß. This model is now intended to make predictions on data with which it has not previously been trained or tested. For this purpose, AIS data from April 28 to May 6 is used. This data is from the same area as the data used to train the model. It is further prepared using the

information on the waterways along the Elbe, the Weser and to and from Wilhelmshaven as described in section 6.1. In total, predictions about the next 5 transition points are made for 932 tracks, of which 242 navigate along the Weser River, 643 along the Elbe River and 47 navigate to and from Wilhelmshaven. Next, the predicted features for each transition point and the true feature are compared to see if the defined threshold for that feature has been exceeded. Should this be the case, the track is flagged.

As noted, in certain cases the model can predict a negative distance to the starboard buoy. Since this would mean that the vessel is sailing outside the fairway, these predictions are erroneous. In these cases, a d_s of 0 is assumed instead of the negative distance.

The results of the marked transition points are shown in table 16. For COG no anomalous tracks were flagged, therefore it is not listed in the table.

	Overall	$\boldsymbol{d_s > 846}$	**%**	$\boldsymbol{SOG > 6}$	**%**	$\boldsymbol{ß > 46}$	**%**
Weser	242	0	0	3	1.2	6	2.5
Elbe	643	5	0.8	3	0.5	0	0
WHV	47	0	0	0	0	0	0
Overall	932	5	0.5	6	0.6	6	0.6

16 Results Anomaly Detection

As can be seen, the number of overall marked tracks are quite similar for each feature under consideration. While 0,6% of the total tracks are marked by a deviation at ß and SOG, 0,5% for d_s and none is marked by the deviating from COG. The marked tracks are then again divided into the individual waterways considered, whereby it is noticeable that when considering d_s, none of the tracks along Wilhelmshaven and along the Weser are marked. For SOG, none track is marked for Wilhelmshaven, 0,5% for the Elbe and 1,2% along the Weser. When considering ß, it is noticeable, that only tracks along the Weser (2,5%) are marked.

Overall, the marked tracks would need to be investigated further to determine if the anomalies detected were actual deviations vessel's usual route or whether they were unrealistic predictions of the model. However, this is outside the scope of this thesis. Nevertheless, the assumption can be made that these marked tracks are related to the outliers of the predictions mentioned in the previous section.

7 Summary and Outlook

At the beginning of this thesis, the following three questions were derived from the problem description.

1. How can vessel tracks be predicted by incorporating multiple data sources?
2. To what extent does adding multiple data sources improve prediction accuracy?
3. How can the developed method be used to flag anomalous tracks?

Based on the analyses in this thesis, these questions can now be answered. To answer the first question, a concept predicting transition points which represent vessel tracks inside waterways was successfully developed and it combined, historical vessel data in form of AIS data with weather information and tide data. Positions of buoys were extracted from see chart information and were also combined with AIS data to create transition points. The data resulting from this concept was then used to train an Bi-LSTM and a transformer model. These models can be used to iteratively predict the subsequent transition points that represent the track a vessel will take. In summary, a concept was developed to predict vessel tracks by incorporating multiple data sources.

Based on the concept developed, the second question can also be answered. Upon evaluating the vessel track predictions, it turns out that adding tide data and weather information does not improve the prediction results in general. Only the transformer model benefited from the addition of tide data and predicted more accurate tracks. In the approach considered, the model that predicts the most accurate vessel tracks was a Bi-LSTM model trained without tide and weather information, focusing only on AIS data combined with position of buoys that delimit waterways. It also turns out that the Bi-LSTM model made more accurate predictions than the transformer model. However, also the transformer model performed better than a linear prediction for three of four features.

The model that made the most accurate predictions was then also tested regarding how successfully anomalous tracks could be flagged. For this purpose, the dataset used to train the model was analyzed and two standard deviations were set as the threshold for anomalous tracks. In addition, another dataset was processed in the same region but at a different time. Having made the most accurate predictions, the Bi-LSTM model was chosen to predict the tracks. As soon as the deviations were greater than the set threshold, they were flagged as anomalous tracks. In summary, the method can detect anomalous tracks by comparing the predictions with the actual measurements. However, it was found that when using the standard deviation of each feature as a threshold, depending on the feature, 0,6% of the tracks were marked as anomalous.

Future work on this topic could aim to make predictions more precise, e.g., by training models for each vessel type specifically. The transformer model could also be improved further. In this thesis, this model overfitted as soon as more layers were added to the model. With a different architecture, this could be prevented, making even more precise predictions possible. The concept could also be applied in other coastal regions to test its usefulness there. In addition, further research could be conducted to determine how tide information and weather data can be incorporated into historical vessel data to make vessel track predictions even more accurate.

8 References

[1] United Nations Conference on Trade and Development, *Review of Maritime Transport 2021*. New York, NY, USA, 2021. [Online]. Available: https://unctad.org/system/files/official-document/rmt2021_en_0.pdf

[2] M. M. Zuzanna Szymanska, *Maersk container ship runs aground off German island.* [Online]. Available: https://www.reuters.com/world/europe/container-ship-runs-aground-off-german-island-2022-02-03/ (accessed: Mar. 29 2023).

[3] T. M. Executive, *Cargo Ship Arrives in Germany with Large Hole After Striking Wind Farm*, 2023. Accessed: May 2 2023. [Online]. Available: https://maritime-executive.com/article/cargo-ship-arrives-in-germany-with-large-hole-after-striking-wind-farm

[4] D. Nguyen and R. Fablet, "TrAISformer-A generative transformer for AIS trajectory prediction," in *CoRR.* [Online]. Available: http://arxiv.org/pdf/2109.03958v1

[5] Oxford University Press, *Anomaly.* [Online]. Available: https://www.oxfordlearnersdictionaries.com/definition/english/anomaly?q=anomaly (accessed: May 9 2023).

[6] R. Yan and S. Wang, "Study of Data-Driven Methods for Vessel Anomaly Detection Based on AIS Data," in *Smart Innovation, Systems and Technologies, Smart Transportation Systems 2019*, X. Qu, L. Zhen, R. J. Howlett, and L. C. Jain, Eds., Singapore: Springer Singapore, 2019, pp. 29–37.

[7] D. Nguyen, R. Vadaine, G. Hajduch, R. Garello, and R. Fablet, "A Multi-Task Deep Learning Architecture for Maritime Surveillance Using AIS Data Streams," in *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, Turin, Italy, 2018, pp. 331–340.

[8] X. Zhang, X. Fu, Z. Xiao, H. Xu, and Z. Qin, "Vessel Trajectory Prediction in Maritime Transportation: Current Approaches and Beyond," *IEEE Trans. Intell. Transport. Syst.*, pp. 1–19, 2022, doi: 10.1109/TITS.2022.3192574.

[9] M. Stróżyna, W. Abramowicz, K. Węcel, D. Filipiak, and J. Małyszko, *Data Analysis in the Maritime Domain*, 2022.

[10] MarineTraffic:, *Global Ship Tracking Intelligence.* [Online]. Available: https://www.marinetraffic.com/en/ais/home/centerx:-12.0/centery:25.0/zoom:4 (accessed: Mar. 9 2023).

[11] International Telecommunication Union., *Recommendation ITU-R M.1371-4. Technical characteristics for an automatic identification system using timedivision multiple access in the VHF maritime mobile band.* [Online]. Available: https://www.itu.int/rec/R-REC-M.1371/en

[12] Tim Stephens, *Global analysis shows where fishing vessels turn off their identification devices.* [Online]. Available: https://news.ucsc.edu/2022/11/unseen-fishing.html (accessed: May 2 2023).

[13] A. Androjna, M. Perkovič, I. Pavic, and J. Mišković, "AIS Data Vulnerability Indicated by a Spoofing Case-Study," *Applied Sciences*, vol. 11, no. 11, p. 5015, 2021, doi: 10.3390/app11115015.

[14] ECMWF, *ECMWF - About.* [Online]. Available: https://www.ecmwf.int/en/about (accessed: Mar. 31 2023).

[15] *About Copernicus | Copernicus.* [Online]. Available: https://www.copernicus.eu/en/about-copernicus (accessed: Mar. 9 2023).

[16] Hans Hersbach *et al.,* "The ERA5 global reanalysis," *Quarterly Journal of the Royal Meteorological Society*, vol. 146, pp. 1999–2049, 2020.

[17] *Dashboard - CMEMS In Situ TAC.* [Online]. Available: http://www.marineinsitu.eu/dashboard/ (accessed: Mar. 9 2023).

[18] M. Steidel, J. Mentjes, and A. Hahn, "Context-Sensitive Prediction of Vessel Behavior," *JMSE*, vol. 8, no. 12, p. 987, 2020, doi: 10.3390/jmse8120987.

[19] International Association of Lighthouse Authorities, *R1001 the IALA Maritime Buoyage System*. Eden Island, Seychelles, 2018.

[20] M. H. Sazli, "A brief review of feed-forward neural networks," *Communications Faculty of Sciences University of Ankara Series A2-A3 Physical Sciences and Engineering*, vol. 50, no. 01, 2006.

[21] Rumelhart, David E. Hinton, Geoffrey E., Williams, Ronald J., "Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations: Learning internal representations by error propagation," in pp. 318–362.

[22] Ian Goodfellow and Yoshua Bengio and Aaron Courville, *Deep Learning*: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org/

[23] Diederik P. Kingma and Jimmy Ba, "Adam: A Method for Stochastic Optimization," in *ICLR*.

[24] S. Hochreiter, “The vanishing gradient problem during learning recurrent neural nets and problem solutions,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[25] R.L.T. Hahnloser, “On the piecewise analysis of networks of linear threshold neurons,” *Neural Networks*, vol. 11, no. 4, pp. 691–697, 1998, doi: 10.1016/S0893-6080(98)00012-4.

[26] Stephen Merity, Nitish Shirish Keskar, and Richard Socher, “Regularizing and Optimizing LSTM Language Models,” in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id= SyyGPP0TZ

[27] Hochreiter, Sepp, Schmidhuber, Jürgen, “Long Short-term Memory,” in *Neural computation*, pp. 1735–1780. [Online]. Available: 10.1162/neco.1997.9.8.1735

[28] Felix Alexander Gers, Jürgen Schmidhuber, and Fred Cummins, “Learning to Forget: Continual Prediction with LSTM,” *Neural Computation*, vol. 12, pp. 2451–2471, 2000.

[29] H. Fan, M. Jiang, L. Xu, H. Zhu, J. Cheng, and J. Jiang, “Comparison of Long Short Term Memory Networks and the Hydrological Model in Runoff Simulation,” *Water*, vol. 12, no. 1, 2020, doi: 10.3390/w12010175.

[30] A. Graves and J. Schmidhuber, “Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks,” in *Proceedings of the 21st International Conference on Neural Information Processing Systems*, 2008, pp. 545–552.

[31] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan, “Show and Tell: A Neural Image Caption Generator,” *CoRR*, abs/1411.4555, 2014.

[32] S. Zhang, L. Wang, M. Zhu, S. Chen, H. Zhang, and Z. Zeng, “A Bi-directional LSTM Ship Trajectory Prediction Method based on Attention Mechanism,” in *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, Chongqing, China, 2021, pp. 1987–1993.

[33] Mike Schuster and Kuldip K. Paliwal, “Bidirectional recurrent neural networks,” *IEEE Trans. Signal Process.*, vol. 45, pp. 2673–2681, 1997.

[34] I. Sutskever, O. Vinyals, and Q. V. Le, *Sequence to Sequence Learning with Neural Networks*: arXiv.

[35] D. Bahdanau, K. Cho, and Y. Bengio, “Neural Machine Translation by Jointly Learning to Align and Translate,” Sep. 2014. [Online]. Available: http://arxiv.org/pdf/1409.0473v7

[36] D. Bahdanau, K. Cho, and Y. Bengio, “Neural Machine Translation by Jointly Learning to Align and Translate,” Sep. 2014. [Online]. Available: http://arxiv.org/pdf/1409.0473v7

[37] A. Vaswani *et al., Eds., Attention Is All You Need,* 17th ed. Red Hook, NY, USA: Curran Associates Inc., 2017. [Online]. Available: https://dblp.org/rec/journals/corr/VaswaniSPUJGKP17.bib

[38] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[39] Seyed Mehran Kazemi *et al.,* “Time2Vec: Learning a Vector Representation of Time,” *CoRR*, abs/1907.05321, 2019.

[40] Q. Wen *et al.,* “Transformers in Time Series: A Survey,” in *arXiv.* [Online]. Available: http://arxiv.org/pdf/2202.07125v3

[41] H. Zhou *et al.,* “Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting,” Dec. 2020. [Online]. Available: http://arxiv.org/pdf/2012.07436v3

[42] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff, “A Transformer-based Framework for Multivariate Time Series Representation Learning,” Oct. 2020. [Online]. Available: http://arxiv.org/pdf/2010.02803v3

[43] S. Mehri, A. A. Alesheikh, and A. Basiri, “A Contextual Hybrid Model for Vessel Movement Prediction,” *IEEE Access*, vol. 9, pp. 45600–45613, 2021, doi: 10.1109/ACCESS.2021.3066463.

[44] D. Gao, Y. Zhu, J. Zhang, Y. He, K. Yan, and B. Yan, “A novel MP-LSTM method for ship trajectory prediction based on AIS data,” *Ocean Engineering*, vol. 228, p. 108956, 2021, doi: 10.1016/j.oceaneng.2021.108956.

[45] C.-H. Yang, C.-H. Wu, J.-C. Shao, Y.-C. Wang, and C.-M. Hsieh, “AIS-Based Intelligent Vessel Trajectory Prediction Using Bi-LSTM,” *IEEE Access*, vol. 10, pp. 24302–24315, 2022, doi: 10.1109/ACCESS.2022.3154812.

[46] C. Liu *et al.,* “TPR-DTVN: A Routing Algorithm in Delay Tolerant Vessel Network Based on Long-Term Trajectory Prediction,” *Wirel. Commun. Mob. Comput.*, vol. 2021, 2021, doi: 10.1155/2021/6630265.

[47] J. Venskus, P. Treigys, and J. Markevičiūtė, “Unsupervised marine vessel trajectory prediction using LSTM network and wild bootstrapping techniques,” *NAMC*, vol. 26, no. 4, pp. 718–737, 2021, doi: 10.15388/namc.2021.26.23056.

[48] D.-D. Nguyen, C. Le Van, and M. I. Ali, “Vessel Trajectory Prediction using Sequence-to-Sequence Models over Spatial Grid,” in *Proceedings of the 12th ACM International Conference on Distributed and Event-based Systems*, Hamilton New Zealand, 2018, pp. 258–261.

[49] P. Dijt and P. Mettes, “Trajectory Prediction Network for Future Anticipation of Ships,” in *Proceedings of the 2020 International Conference on Multimedia Retrieval*, Dublin Ireland, 2020, pp. 73–81.

[50] Nicola Forti, Leonardo M. Millefiori, Paolo Braca, and Peter Willett, *2020 IEEE International Conference on Acoustics, Speech, and Signal Processing: Proceedings : May 4-8, 2020, Centre de Convencions Internacional de Barcelona (CCIB), Barcelona, Spain*. Piscataway, NJ, USA: IEEE, 2020. [Online]. Available: https://ieeexplore.ieee.org/servlet/opac?punumber=9040208

[51] 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 Trans. Aerosp. Electron. Syst.*, vol. 57, no. 6, pp. 4329–4346, 2021, doi: 10.1109/TAES.2021.3096873.

[52] J. Sekhon and C. Fleming, “A Spatially and Temporally Attentive Joint Trajectory Prediction Framework for Modeling Vessel Intent,” in *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, 2020, pp. 318–327. [Online]. Available: https://proceedings.mlr.press/v120/sekhon20a.html

[53] Mengzhen Ding, Wei Su, Yingjie Liu, Jiuwen Zhang, Jianrui Li, Jinzhao Wu, “A Novel Approach on Vessel Trajectory Prediction Based on Variational LSTM: Dalian, China, June 27-29, 2020,” *Proceedings of 2020 IEEE International Conference on Artificial Intelligence and Computer Applications*, 2020, doi: 10.1109/ICAICA50127.2020.

[54] L. You *et al.,* “ST-Seq2Seq: A Spatio-Temporal Feature-Optimized Seq2Seq Model for Short-Term Vessel Trajectory Prediction,” *IEEE Access*, vol. 8, pp. 218565–218574, 2020, doi: 10.1109/ACCESS.2020.3041762.

[55] B. Murray and L. P. Perera, “A dual linear autoencoder approach for vessel trajectory prediction using historical AIS data,” *Ocean Engineering*, vol. 209, p. 107478, 2020, doi: 10.1016/j.oceaneng.2020.107478.

[56] Chao Liu, Yingbin Li, Ruobing Jiang, Yong Du, Qian Lu, Zhongwen Guo, “TPR-DTVN: A Routing Algorithm in Delay Tolerant Vessel Network Based on Long-Term Trajectory Prediction,”

[57] L. M. Millefiori, P. Braca, K. Bryan, and P. Willett, “Modeling vessel kinematics using a stochastic mean-reverting process for long-term prediction,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 52, no. 5, pp. 2313–2330, 2016, doi: 10.1109/TAES.2016.150596.

[58] Fujii, Y., & Tanaka, K., “Traffic Capacity,” *Journal of Navigation*, no. 24, pp. 543–552, 1971. [Online]. Available: doi://10.1017/S0373463300022384

[59] B. Ristic, “Detecting Anomalies from a Multitarget Tracking Output,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 50, no. 1, pp. 798–803, 2014, doi: 10.1109/TAES.2013.130377.

[60] Nicolas Le Guillarme and Xavier Lerouvreur, *2013 16th International Conference on Information Fusion (FUSION 2013): Istanbul, Turkey, 9 - 12 July 2013*. Piscataway, NJ: IEEE, 2013.

[61] P.-R. Lei, “A framework for anomaly detection in maritime trajectory behavior,” *Knowl Inf Syst*, vol. 47, no. 1, pp. 189–214, 2016, doi: 10.1007/s10115-015-0845-4.

[62] M. Vespe, I. Visentini, K. Bryan, and P. Braca, “Unsupervised learning of maritime traffic patterns for anomaly detection,” in *9th IET Data Fusion & Target Tracking Conference (DF&TT 2012): Algorithms & Applications*, London, UK, 2012, p. 14.

[63] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise,” in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 1996, pp. 226–231.

[64] G. Pallotta, M. Vespe, and K. Bryan, “Vessel Pattern Knowledge Discovery from AIS Data: A Framework for Anomaly Detection and Route Prediction,” *Entropy*, vol. 15, no. 12, pp. 2218–2245, 2013, doi: 10.3390/e15062218.

[65] A.-L. J. Giuliana Pallotta, “Data-driven detection and context-based classification of maritime anomalies,” *2015 18th International Conference on Information Fusion (Fusion)*, pp. 1152–1159, 2015.

[66] L. Zhao and G. Shi, “Maritime Anomaly Detection using Density-based Clustering and Recurrent Neural Network,” *J. Navigation*, vol. 72, no. 04, pp. 894–916, 2019, doi: 10.1017/S0373463319000031.

[67] Hersbach, Hans, Bell, Bill, Berrisford, Paul, Biavati, Gionata, Horányi, András, Muñoz Sabater, Joaquín, Nicolas, Julien, Peubey, Carole, Radu, Raluca, Rozum, Iryna, Schepers, Dinand, Simmons, Adrian, Soci, Cornel, Dee, Dick, Thépaut, Jean-Noël, *ERA5 hourly data on single levels from 1940 to present*: Copernicus Climate Change Service (C3S) Climate Data Store (CDS). Accessed: Mar. 29 2023.

[68] Graham Upton and Ian Cook, *A Dictionary of Statistics,* 3rd ed.: Oxford University Press, 2008.

[69] Pekka Kumpulainen, “Anomaly detection for communication network monitoring applications,” 2014.

[70] T. O’Malley *et al., KerasTuner*, 2019. [Online]. Available: https://github.com/keras-team/keras-tuner

[71] Martín Abadi *et al., TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.* [Online]. Available: https://www.tensorflow.org/

[72] F. Chollet and others, *Keras*.

[73] The pandas development team, *pandas-dev/pandas: Pandas*: Zenodo.

[74] Kelsey Jordahl *et al., geopandas/geopandas: v0.8.1*: Zenodo.

[75] J. D. Hunter, “Matplotlib: A 2D graphics environment,” *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007, doi: 10.1109/MCSE.2007.55.

[76] Charles R. Harris *et al.,* “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, 2020, doi: 10.1038/s41586-020-2649-2.

[77] Max Kuhn and Kjell Johnson, *Feature Engineering and Selection: A Practical Approach for Predictive Models.* [Online]. Available: https://bookdown.org/max/FES/ (accessed: May 9 2023).

[78] George H. John, Ron Kohavi, and Karl Pfleger, “Irrelevant Features and the Subset Selection Problem,” in *Machine Learning Proceedings 1994*, William W. Cohen and Haym Hirsh, Eds., San Francisco (CA): Morgan Kaufmann, 1994, pp.

121–129. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9781558603356500234

Appendix

Authors		Mehri et al. [43]	Gao et al. [44]	Yang et al. [45]	Liu et al. [56]	Zhang et al. [32]	Venskus et al. [47]	Nguyen et al. [48]	Dijt et al. [49]	Forti et al. [50]	Capobianco et al. [51]	Sekhon et al. [52]	Ding et al. [53]	You et al. [54]	Nguyen et al. [4]	Steidel et al. [18]
Prediction	Single-Step Prediction				X											
	Track Prediction	X								X		X				
	Trajectory Prediction		X	X		X	X	X	X		X		X	X	X	X
Data Sources	AIS	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	Meteorological data						X									
	Geographical data	X						X	X							X
Prediction Horizon	Short <= 1h	X	X	X	X	X				X		X	X	X	X	
	Medium 1h > 3h		X		X		X				X		X		X	X
	Long >= 3h		X		X		X	X	X						X	

1 Research Vessel Track Prediction

Code 1: **Create AIS-tracks**

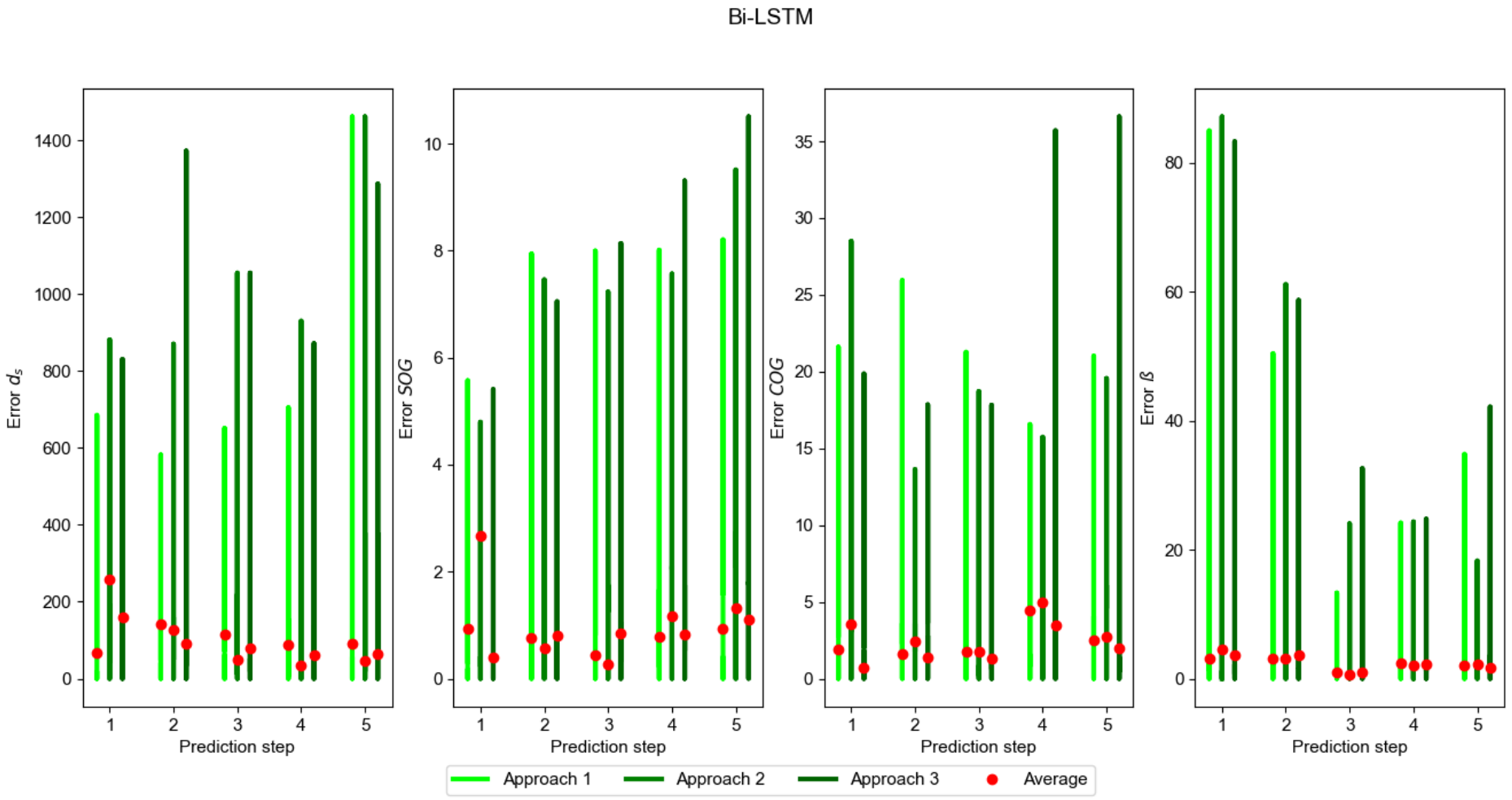
```
'''
Input:
    ais_data: ais-data in the area of interest
    waterway_data: data with positions of the waterway
Output:
    tracks: tracks that contain ais data and follow along the waterway
'''
def create_tracks(ais_data, waterway_data):
    tracks = []
    # filter ais data that is inside the waterway
    filtered_ais_data = find_ais_data_within_waterway(ais_data, waterway_data)
    # group ais data by mmsi
    ais_grouped = filtered_ais_data.grouby("mmsi")
    for group in ais_grouped:
        index = 0
        for i in range(1, len(group)):
            previous_ais_point = group[i-1]
            current_ais_point = group[i]
            # if next ais_point not within one minute of the last ais_point,
            # create new track from index to previous_ais point
            if (current_ais_point -previous_ais_point.time) > minute_1:
                # save track as ais_track
                tracks.append([group, index, previous_ais_point, track_id])
                index = current_ais_point # next track starts from current ais_point
        # save track as ais_track
        tracks.append([group, index, current_ais_point, track_id])
    return tracks
```

2 Pseudo Code: Create AIS-tracks

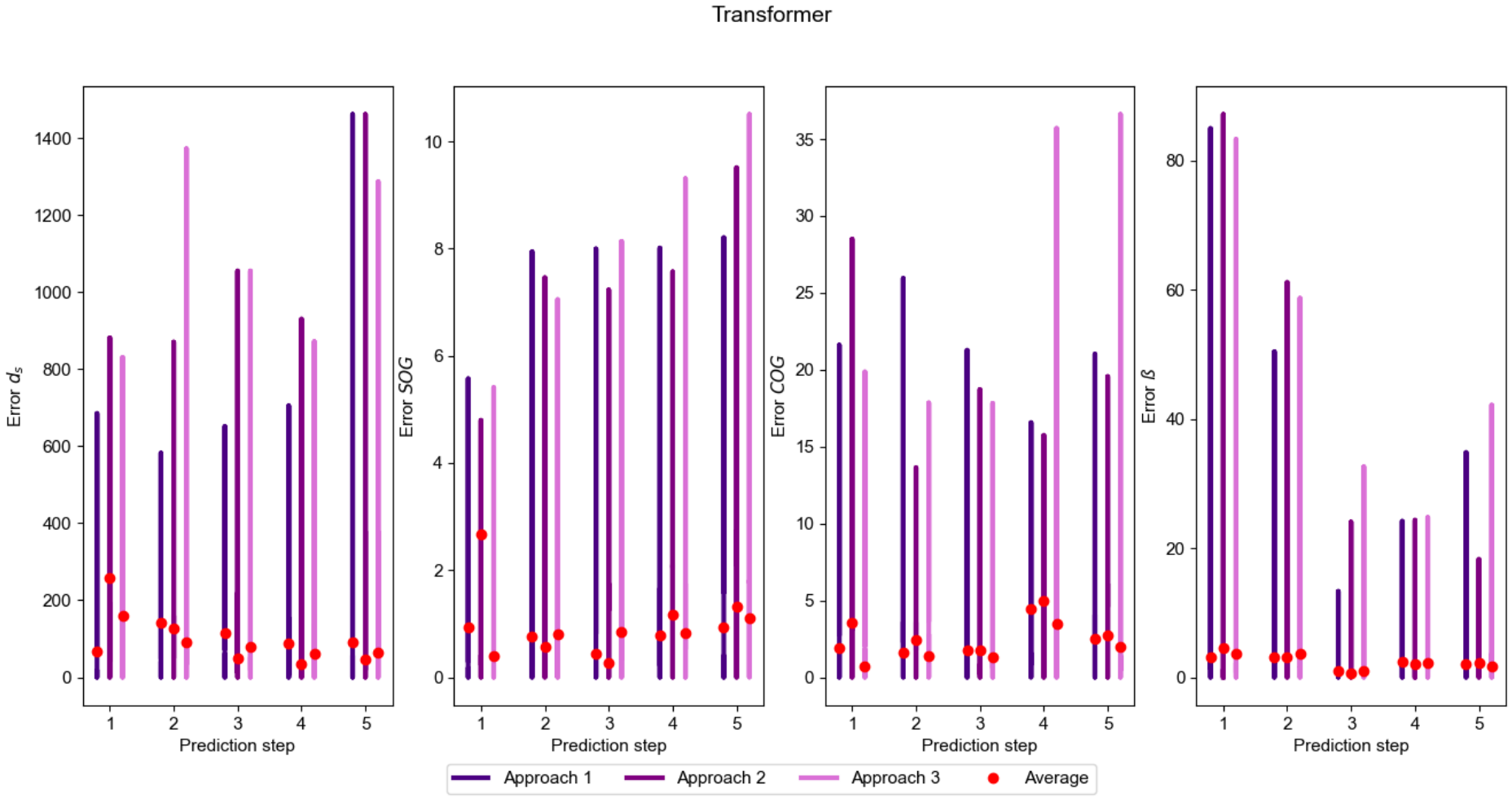
Code 2: **Create Transition-Points**

```
'''
Input:
    ais_tracks: ais_tracks that follow along the waterway
    waterway_data: data with positions of the waterway
Output:
    transition_points: represent transition points inside waterways with the associated
'''
def create_transition_points(ais_tracks, waterway_data):
    transition_points = []
    # group ais data by track_id
    tracks_group = ais_tracks.groupby("track_id")
    for group in tracks_group:
        for i in range(1, len(group)):
            previous_ais_point = group[i-1]
            current_ais_point = group[i]
            # the grid indicates in which waterway grid the point is located
            # only the last point inside the old grid and the first inside the
            # new grid are needed to create transition points
            if previous_point.grid != current_point.grid:
                # calculate the intersection point between the starboard
                # and port buoy of the waterway and the previous and current point
                # which crosses the starboard and port buoy
                inter_p, starboard_buoy = calculate_intersection_point(
                    previous_point, current_point, waterway_data)
                # calculate the distance to te starboard buoy
                distance = starboard_buoy.distance(inter_p)
                sog, cog = interpolate_sog_cog(previous_point, current_point)
                # The points given as arguments build the triangle from which the angle can
                # be calculated current point since it is the first point in the new grid
                ß = calculate_angle(inter_p, starboard_buoy, current_point)
                # add extracted transition point to final list of all transition points
                transition_points.add([distance, sog, cog, ß, track_id])
    return transition_points
```

3 Pseudo Code: Create Transitions-Points



4 Prediction Error Range Bi-LSTM per Step per Feature



5 Prediction Error Range Transformer per Step per Feature

Abschließende Erklärung

Hiermit versichere ich an Eides statt, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Außerdem versichere ich, dass ich die allgemeinen Prinzipien wissenschaftlicher Arbeit und Veröffentlichung, wie sie in den Leitlinien der Carl von Ossietzky Universität Oldenburg festgelegt sind, befolgt habe.

Oldenburg, den 17.05.2023

……

Finn-Mathis Minßen