Contents lists available at ScienceDirect

Reliability Engineering and System Safety

journal homepage: www.elsevier.com/locate/ress

An AIS-based deep learning framework for regional ship behavior prediction

Brian Murray [*], Lokukaluge Prasad Perera

UiT The Arctic University of Norway, Tromsø, Norway

ARTICLE INFO

Keywords:
Maritime safety
Maritime situation awareness
Ship navigation
Trajectory prediction
Collision avoidance
Deep learning
AIS

ABSTRACT

This study presents a deep learning framework to support regional ship behavior prediction using historical AIS data. The framework is meant to aid in proactive collision avoidance, in order to enhance the safety of maritime transportation systems. In this study, it is suggested to decompose the historical ship behavior in a given geographical region into clusters. Each cluster will contain trajectories with similar behavior characteristics. For each unique cluster, the method generates a local model to describe the local behavior in the cluster. In this manner, higher fidelity predictions can be facilitated compared to training a model on all available historical behavior. The study suggests to cluster historical trajectories using a variational recurrent autoencoder and the Hierarchical Density-Based Spatial Clustering of Applications with Noise algorithm. The past behavior of a selected vessel is then classified to the most likely clusters of behavior based on the softmax distribution. Each local model consists of a sequence-to-sequence model with attention. When utilizing the deep learning framework, a user inputs the past trajectory of a selected vessel, and the framework outputs the most likely future trajectories. The model was evaluated using a geographical region as a test case, with successful results.

1. Introduction

Effective maritime traffic monitoring is essential for maintaining the integrity of maritime transportation systems. The safety of human life, as well as that of material assets, and the ocean environment, depend on conducting safe maritime operations. Evaluating the risk associated with maritime transportation systems has been the focus of much research [1–3]. Maritime situation awareness can be argued to be one of the most essential elements with regards to maintaining the safety of such systems. Situation awareness is defined as being aware of what is happening around oneself, and understanding the implications of the current situation now, as well as in the future [4]. All navigators must have an adequate degree of situation awareness to effectively conduct operations at sea. In this context, the primary challenge relates to detecting obstacles and predicting close-range encounter situations. As such, effective collision avoidance can be viewed as a key component of safe maritime transportation systems.

Navigators rely on visual observation, as well as any navigational tools they have available to them, to maintain an adequate degree of situation awareness. Such tools include radar, conning, ECDIS (Electronic Navigation Chart Display and Information System) and AIS (Automatic Identification System). With respect to collision avoidance, navigators rely heavily on radar systems facilitated by ARPA (Automatic Radar Plotting Aid) in addition to the ECDIS. The best navigational tools should be available to navigators to support the navigator in identifying high risk situations [5], such that they can conduct effective collision avoidance maneuvers that adhere to the COLREGS [6]. Generally, collision risk is evaluated for ship to ship encounters, but studies have also addressed quantifying ship collision risk when passing offshore installations [7].

Generally, a linear constant velocity model is utilized to evaluate potential close-encounter situations in order to evaluate the risk of collision. In this manner, the future position of a vessel is predicted using constant speed and course over ground values. This method is reliable, and provides the basis for many commercial systems for predictive traffic surveillance [8]. However, they are inherently constrained by their linearity, and will have degraded performance when predicting complex behavior. More advanced techniques, e.g. [9,10], where extended Kalman filters were utilized, can aid in predicting more complex ship behavior. However, such techniques will not be useful for prediction horizons greater than a few minutes.

Perera and Murray [11] suggested to introduce an advanced ship predictor to aid maritime situation awareness. The predictor is comprised of a local and global predictor to overcome such issues. On a local scale, such techniques can be used to predict short-term ship behavior (order 0–5 min). A global predictor is used to predict more long-term behavior (order 5–30 min). The goal of such global predictions is to prevent close-encounter situations from arising. By predicting the future trajectory of vessels accurately, the future collision risk

[*] Corresponding author.
E-mail address: brian.murray@uit.no (B. Murray).

https://doi.org/10.1016/j.ress.2021.107819

Available online 27 May 2021

Nomenclature

$\mathbf{b}$	Bias Vector
$\mathbf{c}$	Class Vector
$\mathbf{D}_{KL}$	Kullback–Leibler Divergence
$\mathbf{e}$	Principle Component
f	Arbitrary Function
$\mathbf{h}$	Hidden State
J	Loss Function
L	Sequence Length
$\mathbf{n}$	New Candidate Vector
N	Number of Layers
p	Probability
$\mathbf{p}$	Position Vector
q	Approximate Encoder
$\mathbf{r}$	Reset Gate
$\mathbf{s}$	Static Data
$\mathbf{u}$	Update Gate
$\mathbf{v}$	Speed over Ground
$\mathbf{v}$	Input to Softmax Layer
$\mathbf{W}$	Weight Matrix
$\mathbf{x}$	Input Sequence
$\mathbf{y}$	Target Sequence
$\mathbf{z}$	Latent Representation Vector
β	Weighting Hyperparameter
$\boldsymbol{\chi}$	Course over Ground Vector
$\boldsymbol{\mu}$	Mean Vector
$\boldsymbol{\sigma}$	Standard Deviation
$\boldsymbol{\Theta}$	Model Parameters

Subscripts

cat	Categorical
cont	Continuous
h	Hidden
i	Class Number
n	New Candidate
r	Reset
t	State
u	Update
x	Input
z	Latent Representation
$\boldsymbol{\phi}$	Encoder Parameters
$\boldsymbol{\theta}$	Decoder Parameters

Superscripts

$\hat{}$	Estimated Parameter/State
l	Layer

Acronyms

AIS	Automatic Identification System
GRU	Gated Recurrent Unit
HDBSCAN	Hierarchical Density-Based Spatial Clustering of Applications with Noise
KL	Kullback–Leibler
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SAR	Synthetic Aperture Radar
UTM	Universal Transverse Mercator
VAE	Variational Autoencoder
VRAE	Variational Recurrent Autoencoder
VTS	Vessel Traffic Service

between two neighboring vessels can be computed. In this manner, the risk of future close-encounter situations can be predicted, and appropriate collision avoidance actions implemented [12]. Such global behavior may, however, be complex, and will require more advanced techniques to effectively predict.

Developments within maritime traffic monitoring systems can assist in providing situation awareness to navigators, such that proactive collision avoidance maneuvers can be conducted. Vessel Traffic Service (VTS) systems collect traffic data from a variety of sources, including AIS, shore-based radar, Long-Range Identification and Tracking, as well as Synthetic Aperture Radar (SAR) satellite imagery to support maritime traffic safety. The data from such real-time observations are used by VTS operators to support proactive traffic management [8]. The ubiquity of data relating to maritime traffic opens up for opportunities to take advantage of recent developments in machine learning and artificial intelligence.

1.1. Historical AIS data

Historical AIS data provide insight into the historical behavior of ships in given regions, which can be used for maritime traffic data mining and forecasting techniques. Research into utilizing these data to support maritime transportation systems has been the topic of much research recently, with a review of various applications found in [13]. For instance, Montewka et al. [14] utilized AIS data to model the probability of vessel collisions and Goerlandt and Kujala [15] simulated maritime traffic and assessed the probability of collisions. Bye and Aalberg [16] also utilized historical AIS data to conduct statistical analyses of maritime accidents, and developed a model to predict if an accident is related to navigation. Silveira et al. [17] evaluated the ship collision risk off the coast of Portugal, providing a statistical analysis of the traffic separation schemes and evaluated collision risk. Rong et al. [18] also utilized AIS data to characterize maritime traffic and detect anomalies using data mining, and Yu et al. [19] developed a data-driven Bayesian network risk model. A review of methods to assess waterway risk based on AIS data can also be found in [20].

1.1.1. AIS-based ship behavior prediction

Ristic et al. [21] was one of the first to investigate using AIS data for trajectory prediction. The study used a particle filter for ship behavior prediction based on AIS data. The uncertainty of the prediction, however, renders the method of limited use with respect to collision avoidance purposes. A number of studies have also addressed clustering historical AIS trajectories, classifying a vessel to a given cluster and conducting a prediction. Pallotta et al. [22] introduced the TREAD (Traffic Route Extraction and Anomaly Detection) method to cluster historical trajectories into routes, and classify a partial trajectory to one of these routes. Pallotta et al. [23] expanded this work to predict vessel positions for a cluster discovered by TREAD via an Ornstein Uhlenbeck stochastic process. Mazzarella et al. [24] also applied a Bayesian network approach using a particle filter for trajectory prediction. These methods, however, are useful for predictions in the order of hours, and as such of greater benefit for general maritime traffic forecasting, than for collision avoidance purposes. Xiao et al. [25] also presented an approach to forecast traffic 5 to 60 min into the future by extracting waterway pattern knowledge via a lattice-based technique. The method is computationally efficient, and facilitates predictions with an accuracy relevant to assist general maritime traffic forecasting. However, the accuracy may not be sufficient to assist in supporting proactive collision avoidance with respect to encounter situations.

Other methods include [26], which introduced a single point neighbor search method to predict trajectories using historical AIS data. The method, however, does not handle branching waterways, and the accuracy of the method is limited. Dalsnes et al. [27] expanded this approach to provide multiple predictions using a prediction tree. The resultant predictions are then clustered using a Gaussian mixture model. Both these methods, however, do not utilize trajectory clustering prior to conducting a prediction. Predictions are based on the neighborhood of a predicted state, which may include data points that belong to other clusters of ship behavior, inherently degrading the performance.

Rong et al. [28] presented a probabilistic approach to ship behavior prediction using a Gaussian process model. This method had successful results for the investigated region off the coast of Portugal. However, this region did not contain complex traffic situations. Therefore, the outcome of the same approach to more complex traffic regions is inconclusive. Based on a clustering of locally extracted trajectories, Murray and Perera [29] classified a selected vessel to one of the clusters, and predicted the future trajectory using a dual linear autoencoder approach. This approach had successful results, but was computationally expensive with respect to extracting trajectories. This may degrade the results in certain situations with respect to collision avoidance purposes.

1.1.2. Deep learning-based approaches

Machine learning is being applied at an increasing rate in wide variety of domains, including the field of reliability engineering and safety [30]. In [31], for instance, maritime accidents were evaluated using k-means clustering to identify classes of accidents. A sub-field of machine learning known as deep learning [32] has been the center of technological innovation in recent years. With state-of-the-art performance in image and speech recognition [32], the methods have slowly begun to gain the attention of other domains e.g. for predictive maintenance [33]. Within the maritime domain, however, there is still limited research on adopting deep learning techniques.

AIS data are an ideal data set to apply deep learning techniques in the maritime domain. Zhang et al. [34], for instance, applied a convolutional neural network to classify regional ship collision risk levels. Nguyen et al. [35] also developed a multi-task deep learning architecture for maritime surveillance based on a variational recurrent neural network. The framework can be utilized for multiple purposes including trajectory prediction. However, the method applied a 4-hot encoding to the data that reduces the resolution of the predictions, degrading the performance with respect to collision avoidance purposes. These techniques were further developed in [36], where GeoTrackNet was presented to facilitate maritime anomaly detection.

Learning deep representations provide a learned subspace of representations of the input data, and have been used for various goals e.g. transfer learning for remaining useful life prediction [37]. Yao et al. [38] investigated clustering AIS trajectories using deep representation learning, where the results indicated that the deep learning approach outperformed non-deep learning based approaches. Murray and Perera [39] expanded this work, where it was found that a variational recurrent autoencoder architecture provided better representations for trajectory clustering. These methods, however, do not provide a method to predict the future trajectory of a selected vessel. Forti et al. [40] and Capobianco et al. [41] utilized a recurrent neural network to predict trajectories using a sequence-to-sequence model. Such sequence-to-sequence models are in essence encoder–decoder models, and have been used for a variety of applications e.g. predicting software reliability [42]. The results from [40] were promising, but the model has only been tested on a data set of limited complexity. If applied to an entire region of historical data, the performance will likely be degraded.

1.2. Contribution

In this study, it is suggested to utilize historical AIS data to predict ship behavior on a global scale, with the purpose of aiding in proactive collision avoidance. As such, this study investigates predicting the future 30 min trajectory of a selected vessel. In this manner, the safety of maritime transportation systems can be enhanced. It is, therefore, assumed that future ship behavior can be predicted based on the historical behavior of other vessels in a given geographical region. If successful, such methods can aid in providing situation awareness to navigators and VTS centers. Furthermore, such methods can contribute towards risk models for future autonomous vessels [43].

The study presents a deep learning framework for regional ship prediction. Given the past trajectory of a selected vessel, the framework predicts its future trajectory. To facilitate this, the data for a specific geographical region are used to generate a prediction model for ship behavior within this region. Similar methods train neural networks on all the available AIS data. However, for the purpose of aiding in collision avoidance, trajectory predictions should be as accurate as possible. As a result, this study suggests decomposing the historical ship behavior into local models.

To create these local models, it is suggested to cluster historical ship behavior using a variational recurrent autoencoder, as outlined in [39]. This approach is expanded to add more complexity to the model, resulting in improved clustering performance. The method is able to discover clusters of ship behavior, such that local models can be trained for each individual cluster. Such local models should have enhanced performance, as they are trained on specific ship behavior. In contrast, training on all available data will result in models that must capture a much larger degree of variation, inherently degrading their performance due the increased complexity of the underlying data.

The method further suggests to classify a trajectory segment to a given cluster using a deep learning architecture. The method outputs a distribution over possible clusters of behavior the trajectory may belong to, such that multiple predictions can be made. It is highly likely that the trajectory belongs to one of these clusters, and as a result, one of the trajectory predictions should be accurate.

The local models are trained on the data in each unique cluster of historical behavior. In this study, a sequence-to-sequence model using an attention mechanism is suggested to function as the local model for each cluster. Such attention mechanisms provide the basis for state-of-the-art translation architectures, improving the performance significantly compared to conventional sequence-to-sequence models. Furthermore, sequence-to-sequence models should have enhanced performance compared to standard recurrent neural networks, as they predict an entire sequence based on an entire input sequence. As such, the error for the entire future sequence is used to optimize the network, and not just error for one time step at a time.

The overall framework outlined in this study provides a novel contribution to conduct efficient trajectory predictions. Using pre-trained networks for a given geographical region, the most probable future trajectories can be output to a user in under a second.

2. Methodology

In this section, the proposed methodology of the deep learning framework is outlined. The framework is designed such that it can be applied to any geographical region. The objective of the framework is to support ship behavior prediction. It is assumed that the respective vessels observed in a given geographical region may have similar behavior to that of other vessels in the past. By developing a framework to model the historical behavior of the respective ships for a given region, it may be possible to predict the future behavior of a selected vessel. This is achieved through the use of historical AIS data.

An overview of the framework is illustrated in Fig. 1. Overall, the framework can be viewed as being conducted in two phases. The first

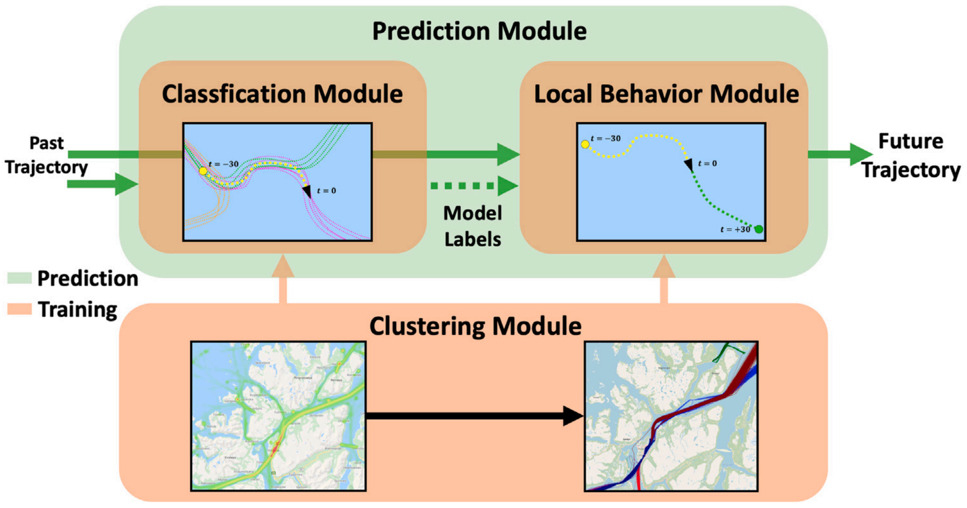

Fig. 1. Overview of deep learning framework.

is the training of the modules, where the trained models are illustrated in orange in Fig. 1. The second phase is the prediction phase, and uses the pre-trained networks. This phase is illustrated in Fig. 1 in green.

The clustering module is trained first using all available historical AIS data for the selected geographical region. The goal of the clustering module is to discover clusters of historical ship behavior. These clusters contain historical AIS trajectories that have similar behavior. Fig. 1 depicts the clustering module in orange. The left figure in the module presents a heat map of the available historical AIS data for a specified region, and the right a subset of clusters of ship behavior. These clusters correspond to groupings of similar historical ship behavior. Such clusters may, for instance, involve alternate routes, or speed profiles along routes.

The purpose of the prediction module is to predict the future trajectory of a selected vessel, given its observed past behavior. It is assumed in this study that the past 30 min of AIS data are available. The input to the prediction module, as illustrated in Fig. 1, corresponds, therefore, to the past 30 min behavior of a selected vessel. However, the architecture can be trained based on any input trajectory length.

The prediction module consists of two sub-modules, the classification module and local behavior module. In the classification module, the input trajectory is matched to one of the behavior clusters discovered in the clustering module. The input trajectory in this study is the past 30 min behavior of the selected vessel. The classified cluster label is then input to the local behavior module, which selects the pre-trained model that corresponds to that cluster of behavior. This model is then used to predict the future 30 min behavior of the selected vessel. The classification module also outputs multiple possible clusters the trajectory may belong to, with a probability associated with each cluster. In this manner, multiple trajectories can be predicted based on the local models for the classified behavior clusters.

2.1. Preprocessing

Prior to training the neural networks involved in this study, preprocessing of the AIS data must be conducted. The first step is to generate complete trajectories from the unprocessed AIS data. This is conducted by extracting trajectory segments, where the time between consecutive points exceeds some parameter. In this study, trajectories are defined where any two points are more than 30 min apart. Furthermore, each individual trajectory is interpolated at one minute intervals to facilitate higher density data, as well as provide a common foundation for training the network. As such, each vessel state, $\mathbf{x}_t$, will be one minute apart. Each vessel state is defined in (1), and contains the positional data defined in UTM coordinates $\mathbf{p} = [p_1, p_2]$, as well as the speed

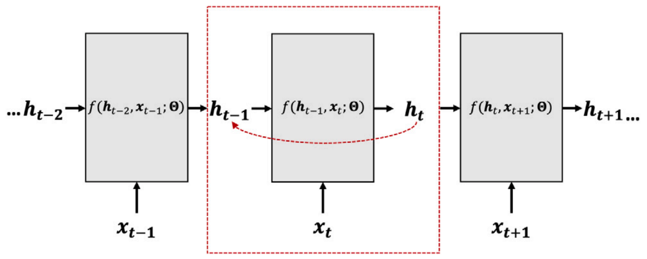
...h_{t-2}
$f(h_{t-2}, x_{t-1}; \Theta)$
h_{t-1}
$f(h_{t-1}, x_t; \Theta)$
h_t
$f(h_t, x_{t+1}; \Theta)$
h_{t+1}...
x_{t-1}
x_t
x_{t+1}

Fig. 2. RNN.
Source: Illustration adapted from [39].

over ground, v, and course over ground, χ, decomposed into the UTM coordinate directions, $\chi = [\chi_1 \ \chi_2]$.

$$\mathbf{x}_t = [p_1, p_2, v, \chi_1, \chi_2] \tag{1}$$

However, the parameters in the vessel states vary significantly in magnitude. As such, all states are scaled across each parameter for all extracted trajectories from the region of interest. In this case, the values are scaled between $[-1, 1]$ given that the data are more optimal for the recurrent neural networks that make use of the tanh function.

All trajectories are present in the input data, i.e. no anomalous trajectories have been removed from the data set. This is due to the ability of the clustering module to identify such trajectories, and remove them before further processing. This is addressed in Section 2.3.

2.2. Recurrent neural networks

Recurrent neural networks (RNNs) [44] are designed to handle sequence data. The general RNN architecture is visualized in Fig. 2. As historical AIS trajectories are multivariate time series, RNNs are chosen to serve as the main deep learning architecture in the framework utilized in this study. RNNs are capable of handling time series of variable length, and can be combined with other architectures to achieve various goals. RNNs are ideal for time series data in that they incorporate a sense of memory into the network. Given a time series $\mathbf{x} = \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_L\}$ of length L, a recurrent neural network processes the input state $\mathbf{x}_t$ at a given state, t, sequentially. In addition, information about the time series prior to state t is processed through the previous hidden state, $\mathbf{h}_{t-1}$. The network then outputs the current hidden state, $\mathbf{h}_t$, that incorporates relevant information from $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$ in (2).

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t; \boldsymbol{\Theta}) \tag{2}$$

Each operation can be thought of as applying (2) in an RNN cell. The same operation repeats for all states, and is in this sense recurrent. A recurrent neural network can be thought of as an unfolded computational graph, where each operation, i.e. each cell, applies (2). In this manner the parameters are shared between all operations. The recurrence is visualized within the red box in Fig. 2, indicating that each cell is fed the previous cells output along with the current input. The architecture is in this sense causal, where the current output depends on all the past time steps. It is evident that such an architecture is applicable to ship trajectories, in that the future behavior should be dependent on the past behavior.

2.2.1. Gated Recurrent Unit

The original RNN architecture is often referred to as the vanilla RNN. When training this architecture, the network struggles to learn long-term dependencies. This is due to vanishing gradients during backpropagation of the network [45]. The long-term memory of such networks is, therefore, poor, and can degrade their performance when long-term dependencies in the data exist. The Gated Recurrent Unit (GRU) [46,47] is an recurrent architecture that introduces the concept of gates to reduce the effect of vanishing gradients. Other gated architectures include the Long Short-Term Memory (LSTM) [48]. The GRU, however, reduces the number of model parameters compared to the LSTM, thereby reducing training time.

2.2.2. Bidirectional RNNs

Standard RNNs conduct calculations in the forward direction, i.e. from the past to the future. Bidirectional RNNs [49], however, provide an architecture where the calculations are conducted in both the forward and backward directions concurrently. In this manner, future events can be thought to affect past events. In the case of ship trajectories, this may not be as intuitive. However, the argument can be made that choices made by a navigator may depend on future choices, e.g. speed changes dependent on a future course alteration, route choice, etc. As such, a bidirectional RNN will incorporate more information about the navigational patterns of past ships in a historical AIS data set.

2.2.3. Stacked RNNs

Deep neural networks, i.e. with multiple layers, have been shown to have superior performance to more shallow networks. The same can be said for RNNs, as it was shown in [50] that increasing depth of RNNs enhanced their performance. Such RNNs are often referred to as stacked RNNs. The stacked architecture implies that there are multiple RNNs that feed into each other as illustrated in Fig. 3. The figure illustrates a network of N layers with 0 being the initial layer and arbitrary layer l between.

2.3. Clustering module

In the clustering module, clusters of ship behavior are discovered. This is achieved through an unsupervised learning technique known as clustering, where the underlying groupings in the data are discovered. The groupings in the case of this study correspond to sets of trajectories with similar behavior. Discovering such groupings, however, can be challenging. Standard clustering techniques require representations of the data to be vectors of equal size. A clustering algorithm will then group the vectors based on some similarity, i.e. distance, measure. Historical AIS trajectories, however, consist of multivariate time series of variable length. As a result, they cannot be clustered using standard techniques. It is, therefore, of interest to develop a framework to generate fixed size representations of the trajectories, such that standard clustering techniques can then be applied to the representations.

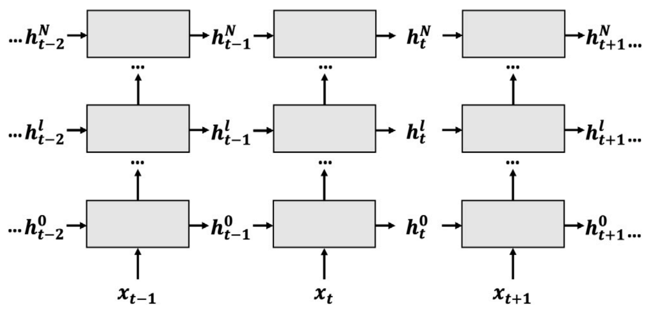

Fig. 3. Stacked RNN with N layers.

Murray and Perera [39] suggested to utilize a deep representation learning-based approach to facilitate trajectory representation generation for subsequent clustering. The study argues that RNNs are ideal for such a task, as they are designed to generate representations of multivariate sequences via their hidden states. The study compares utilizing a recurrent autoencoder and β-variational recurrent autoencoder (β-VRAE) [51] to learn good representations of the data. It was found that the β-VRAE provided more compact groupings, resulting in a more effective clustering scheme. The Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) algorithm [52] was utilized to cluster the trajectory representations with good results.

The method is expanded in this study, where a bidirectional stacked VRAE architecture is utilized to generate representations of the data, which are subsequently clustered using HDBSCAN. The details of the architecture of the clustering module are presented in the following sections.

2.3.1. Hierarchical Density-Based Spatial Clustering of Applications with Noise

The Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) algorithm [52] is utilized to cluster the latent representations from the VRAE. The goal is to identify class labels that can be utilized in the classification module, where each class relates to a cluster of ship behavior. HDBSCAN is a non-parametric clustering approach that can identify clusters of varying density and shape, and was argued in [39] to be powerful in clustering ship trajectory representations generated by a VRAE. This is likely due to irregular grouping of the data in this subspace. In certain cases, the autoencoder generates highly compact groupings of data. This is due to highly similar trajectories of very specific behavior. Such clusters will be very dense. In other cases, less similar trajectories of more general behavior are discovered. These clusters are, therefore, less dense. The groupings are also of highly irregular shape.

HDBSCAN provides a flexible algorithm that is able to discover clusters of varying density and shape. Furthermore, it is able to discover the most likely number of clusters without explicit input. Other algorithms, e.g. k-means, will have degraded performance on such a data set. Such algorithms are unable to capture clusters of the varying shapes and densities as in this study and will, therefore, likely discover a clustering scheme that is not physically meaningful when applied in this subspace. Furthermore, the number of clusters must be input to such an algorithm. This is difficult to estimate for data sets such as those in this study.

The Density-Based Spatial Clustering of Applications with Noise algorithm is extended in HDBSCAN by adapting it to a hierarchical clustering scheme. The algorithm defines core distances for each point as the distance to the kth nearest neighbor. These distances function as local density estimates, and provide the basis for a mutual reachability metric between two points. This metric then provides the basis for a minimum spanning tree and hierarchy. The tree is then pruned using the minimum cluster size, where any clusters below a given threshold are filtered out. The algorithm then discovers the most stable clusters in the hierarchy. Furthermore, HDBSCAN provides the capability to discover noise in the data, where any data points that do not belong to the clusters are labeled as noise. For further details see [52].

The clusters discovered by HDBSCAN represent the regular ship behavior in the region of interest. The data clusters can, therefore, be used to create local models that describe the regular behavior for each cluster. The algorithm also functions as a form of preprocessing, where anomalous trajectories will be labeled as noise, as they do not correspond to any cluster of regular ship behavior. Predicting such anomalies is difficult, as the behavior of such vessels is often highly erratic. This study, therefore, focuses on modeling regular ship behavior, and discards the noise identified by HDBSCAN.

2.3.2. Variational recurrent autoencoder

The goal of the VRAE is to generate meaningful representations of the historical AIS trajectories, such that they can be effectively clustered. A common method to generate meaningful representations is the autoencoder. An autoencoder is comprised of two parts, and encoder and a decoder. The encoder encodes the data to a latent representation, and the decoder subsequently attempts to reconstruct the data from this latent representation.

RNNs inherently provide a compression of the data, where feeding a sequence into an RNN, the network outputs a final hidden state, $\mathbf{h}_L$, that represents the entire input sequence. By training an encoder RNN that encodes the historical ship trajectories to a hidden state, $\mathbf{h}_L$, a decoder can be trained to reconstruct the input trajectory from $\mathbf{h}_L$. Such an architecture is known as a recurrent autoencoder [53]. This is in essence an application of sequence-to-sequence models [54] that provide the basis for the state-of-the-art in natural language processing tasks e.g. translation [46].

The VRAE is extension of the recurrent autoencoder that utilizes a variational autoencoder (VAE) [55,56]. The VAE introduces a probabilistic approach to the autoencoder, where it is assumed that data are generated by a random process from a continuous latent variable denoted $\mathbf{z}$. An approximate probabilistic encoder, $q_\phi(\mathbf{z}|\mathbf{x})$, produces a distribution over the latent variable, $\mathbf{z}$, and a decoder $p_\theta(\mathbf{x}|\mathbf{z})$ reconstructs $\mathbf{x}$ from $\mathbf{z}$. It is, furthermore, assumed that $q_\phi(\mathbf{z}|\mathbf{x})$ is a multivariate Gaussian with a diagonal covariance in (3).

$$q_\phi(\mathbf{z}|\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}_z, \boldsymbol{\sigma}_z^2\mathbf{I}) \tag{3}$$

Both the encoder, $q_\phi(\mathbf{z}|\mathbf{x})$, and decoder, $p_\theta(\mathbf{z}|\mathbf{x})$, are approximated by neural networks. The advantage of using a VAE compared to a standard autoencoder, is that it encourages the latent variables to become normally distributed. Murray and Perera [39] argued that this limits the chaos in the latent space, and encourages the latent representations of the data to be more compact, thereby providing better representations for a clustering algorithm.

Fabius and van Amersfoort [57] extended the VAE to introduce a recurrent architecture in the variational recurrent autoencoder (VRAE). Here the encoder and decoder are comprised of RNNs. An overview of the architecture in this study is presented in Fig. 4. To the left in the figure is the encoder. The encoder is bidirectional, where the forward encoder is illustrated in yellow, and the backward encoder illustrated in blue. Both encoders are GRUs. Integrating a bidirectional architecture, more information can be encoded in the latent space. Furthermore, the bidirectional encoder is stacked, providing increased depth to the network. This allows it to learn more complex relationships in the data. The output of the forward and backward encoders are concatenated to comprise the final hidden state, $\mathbf{h}_L$, of the encoder. This is visualized in orange in Fig. 4. The mean and standard deviation of the normal distribution in (3) are estimated via linear layers in (4) and (5).

$$\boldsymbol{\mu}_z = \mathbf{W}_\mu\mathbf{h}_L + \mathbf{b}_\mu \tag{4}$$

$$\boldsymbol{\sigma}_z = \mathbf{W}_\sigma\mathbf{h}_L + \mathbf{b}_\sigma \tag{5}$$

Using the re-parametrization trick to allow for backpropagation, the latent variable is estimated in (6), where $\boldsymbol{\epsilon}$ is sampled from a normal distribution according to $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$. This is illustrated in purple in Fig. 4.

$$\mathbf{z} = \boldsymbol{\mu}_z + \boldsymbol{\sigma}_z \odot \boldsymbol{\epsilon} \tag{6}$$

The decoder, illustrated in green in Fig. 4, takes the initial hidden state as input. This is calculated in (7).

$$\mathbf{h}_{in} = \tanh(\mathbf{W}_{zh}\mathbf{z} + \mathbf{b}_{zh}) \tag{7}$$

It then reconstructs the input sequence sequentially, where the next state is estimated according to (8).

$$\hat{\mathbf{x}}_{t+1} = \mathbf{W}_{h\hat{x}}\mathbf{h}_t + \mathbf{b}_{h\hat{x}} \tag{8}$$

Each predicted state is fed into the following cell to predict the next. The basis for the entire prediction is the input from $\mathbf{h}_{in}$. Therefore, all the information contained in the sequence must be stored in the latent vector $\mathbf{z}$. Training this encoder–decoder architecture forces the network to learn a meaningful representation of the data in the latent space.

The network is optimized by maximizing a variational lower bound on the log-likelihood (9).

$$J(\theta,\phi;\mathbf{x},\mathbf{z}) = \mathbf{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log(p_\theta(\mathbf{x}|\mathbf{z}))\right] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})) \tag{9}$$

The first term in (9) can be viewed as the reconstruction loss, and is evaluated in this study using the mean squared error. The second term is the Kullback–Leibler (KL) divergence between the approximate posterior, $q_\phi(\mathbf{z}|\mathbf{x})$ (i.e. encoder), and the prior $p_\theta(\mathbf{z})$. In this study is assumed that the prior is normally distributed according to $p_\theta(\mathbf{z}) \sim \mathcal{N}(0, \mathbf{I})$. Therefore, maximizing this second term implies minimizing the KL-divergence. By applying this constraint to the latent space, this term tries to enforce compact groupings of data. Therefore, similar ship trajectories should be encouraged to be closer together in the latent space. As such, a clustering algorithm should be more successful in discovering clusters of trajectories using such an architecture. For further details on representation learning for trajectory clustering, please see [39], as well as [55,56] for further details on VAEs and [57] for VRAEs.

2.4. Classification module

In this study, the aim is to classify 30 min trajectory segments to one (or more) of the discovered clusters. These trajectory segments represent the past trajectory of a selected vessel for the case of a prediction. However, seeing as the autoencoder is trained on regional trajectories, the structure will not be conducive with the 30 min trajectory segments. As a result, the latent representations of such trajectory segments will not be meaningful in relation to the discovered clusters in the same subspace. A classification network must, therefore, be trained to match such 30 min trajectory segments to one (or more) of the discovered clusters.

Each trajectory in the data set is, therefore, split into 30 min segments using a sliding window technique with a one minute interval. In this manner, the classification module will be trained using all possible 30 min trajectory segments. Each segment is assigned a class label corresponding to the class of its parent trajectory, discovered via the clustering module. The objective of the classification module is to correctly classify an input trajectory segment to one of the underlying ship behavior clusters in the data set.

The architecture of the classification module is illustrated in Fig. 5. Given that the trajectory segments consist of sequence data, it is suggested to utilize RNNs to encode the dynamic data. A bidirectional, stacked encoder is, therefore, utilized to encode the trajectory segments to a fixed size vector in a similar manner to the clustering module. The final hidden states of the backward and forward encoders are

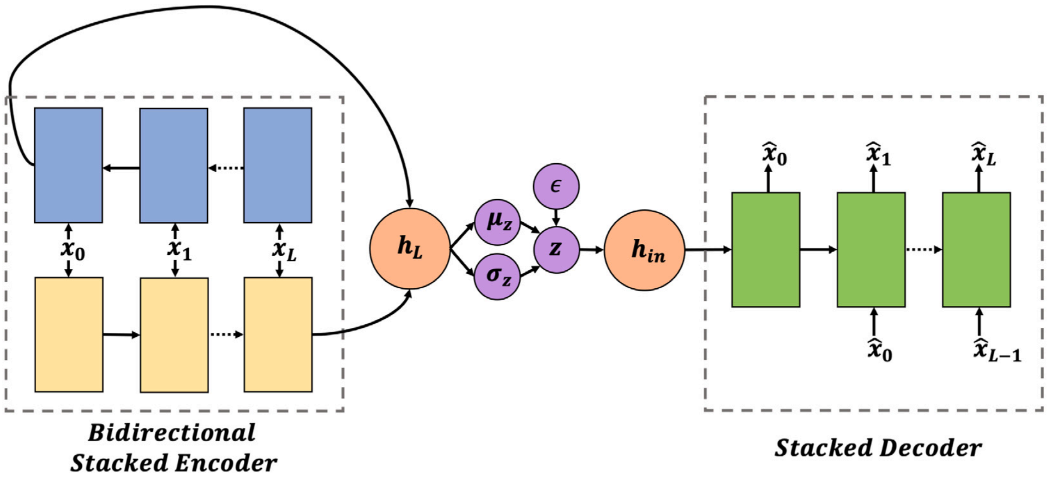

Fig. 4. VRAE with a bidirectional stacked encoder, and stacked decoder. The forward encoder is illustrated in yellow, and the backward encoder in blue. The decoder is illustrated in green. All RNNs are stacked.

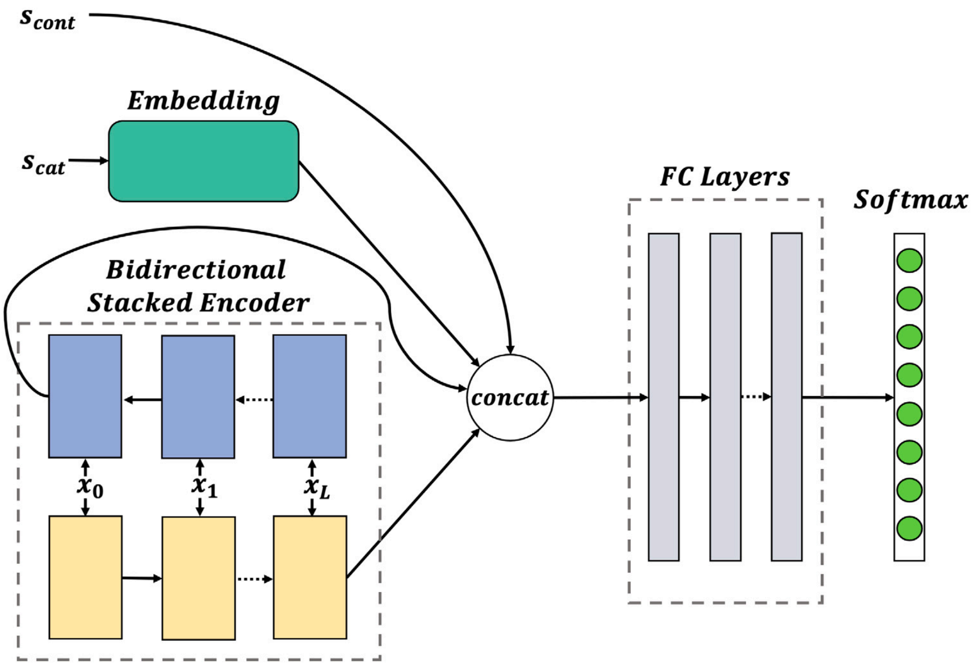

Fig. 5. Classifier with a bidirectional stacked encoder for dynamic data. The embedding block embeds categorical static data. The encoded dynamic data are concatenated with the embedded categorical static data and continuous static data. Fully connected (FC) layers predict the class via a softmax ouput layer.

concatenated, allowing the encoder to preserve dependencies in both directions.

Historical AIS data, however, is not limited the dynamic data represented by the trajectory data. Static data are also available, e.g. ship type, ship length, date, etc. Assuming that this information is available via the AIS at the time of prediction, such static data can be utilized by the classifier to discriminate between classes. In this study, it is suggested to include the ship type and length in the static data. Ship type will play a significant role in the behavior of a vessel, and should aid the classifier in achieving higher accuracy. The ship length should also play a role in the type of behavior to be expected by the ship.

The static data are further separated into categorical data, $\mathbf{s}_{cat}$, and continuous data, $\mathbf{s}_{cont}$. The continuous features, e.g. length, can simply be scaled and concatenated with the dynamic data, as shown in Fig. 5. The categorical data, however, e.g. ship type, must be encoded using an embedding layer. An embedding layer maps a category to a vector representation. The concept was introduced to aid natural language processing, where word embeddings [58] provide the basis for many natural language processing tasks. In this study, it is suggested to embed the categorical static data, and concatenate these embeddings with the remainder of the data as shown in Fig. 5.

The concatenated data are then fed into fully connected, i.e. linear, layers. The final layer will have an output dimension corresponding to the number of classes, i.e. clusters, discovered by the clustering module. A softmax layer then computes the final output by scaling the output between 0 and 1 according to (10).

$$\hat{\mathbf{c}}_i = \frac{e^{\mathbf{v}_i}}{\sum_{j=1}^{C} e^{\mathbf{v}_j}} \tag{10}$$

$\mathbf{v}_i$ is the predicted value for class i from the fully connected layers, and $\hat{\mathbf{c}}_i$ is the softmax output for class i. In this manner, a probability distribution is created over the number of classes. The classifier then compares the softmax output to the true class vector $\mathbf{c}$, where $\mathbf{c}_i = 1$ for the true class and $\mathbf{c}_j = 0 \ \forall \ j \neq i$. The cross entropy loss is then calculated and used to optimize the network.

When training the model, modern deep learning architectures, e.g. PyTorch [59], include a softmax layer in the cross entropy loss. As a result, when training a network using the built in cross entropy loss, the softmax layer is not included in the architecture of the network. When evaluating the model, the predicted class is, therefore, generally taken as the argmax of $\mathbf{v}$ without using a softmax layer. However, given that the softmax function gives a probability distribution over the number of classes, this can be used to identify a distribution over the ship behavior clusters the trajectory segment belongs to. As such, multiple trajectory clusters can be identified as possible for the selected vessel during a prediction. Therefore, a softmax layer is applied to the network during evaluation in this study.

2.4.1. Local behavior module

Given that the clustering module has discovered clusters of ship behavior in the historical AIS data, local models can be created to predict the ship behavior. Each cluster represents a group of localized ship behavior. Training a model on the subset of data corresponding to this local behavior should improve the predictive capabilities of the algorithm, as opposed to training on all available data.

In this study, each local model is comprised of a sequence-to-sequence model [54]. The VRAE in Section 2.3.2 is a such an architecture, where a sequence-to-sequence model is utilized to aid in clustering. As outlined in Section 2.3.2, this encoder–decoder approach is common in natural language processing. The core of such models is an RNN, where the RNN used in this study is a GRU. The encoder RNN encodes the input sequence to a fixed size vector, and the decoder RNN decodes the target sequence using this vector, i.e. latent representation, of the input sequence. In an autoencoder architecture, as in Section 2.3.2, the target sequence is equal to the input sequence. In this manner the decoder's task is to reconstruct the input.

The local models in this study, however, take the past 30 min behavior of a selected vessel as input, and predict the future 30 min behavior. As such, the past 30 min must be encoded into a fixed size vector, and the future 30 min must be predicted using this representation. For an autoencoder, this bottleneck in the latent representation provides the basis for clustering, as one wishes to discriminate between classes in this space. When a sequence-to-sequence model is used for predictions, however, this bottleneck is detrimental to the performance.

The bottleneck in sequence-to-sequence models limits the capacity of the model, as an entire sequence must be predicted from a single vector. Furthermore, the encoder often becomes gradient starved. This is due to the fact that gradients calculated from the loss in the decoder must flow via the bottleneck during backpropagation. As a result, the encoder side of the network does not update well during training. Bahdanau et al. [60] introduced an attention mechanism that addresses this issue. Instead of only looking at the final hidden state of the encoder, the decoder is able to look at all of the encoder hidden states, enhancing the predictive performance of the model. Using such an architecture, gradients are allowed to flow freely to the encoder side of the network via the attention mechanism. This approach was developed for translation tasks, but the architecture is also relevant for sequence-to-sequence tasks involving time series data.

The local model architecture in this study, therefore, utilizes the attention mechanism in [60] to facilitate effective ship trajectory prediction. The architecture of the local model is illustrated in Fig. 6. The attention mechanism is facilitated by a fully connected network, represented by the pink box in Fig. 6. The attention mechanism takes the previous hidden state of the decoder, i.e. $\mathbf{h}_{t-1}$, as well as all of the encoder hidden states as input. In this study, the encoder is bidirectional and stacked. As a result, the input to the attention mechanism will be the hidden states from the top layer, which contain the concatenated backward and forward hidden states of the encoder for each time step. The attention mechanism in this study functions in two steps. First, $\mathbf{h}_{t-1}$ of the decoder is matched with the encoder hidden states. For the case of ship trajectory prediction, this can be thought of as how relevant ship behavior at some point during the past 30 min is for conducting a prediction at the current time step. The network can in this manner learn what to look at in order to most effectively conduct a prediction. The outputs are then run through a softmax layer as in (10). This generates an attention distribution, $\mathbf{a}$, over the encoder hidden states, where each attention value can be viewed as a weight for the corresponding encoder hidden state. A weighted sum of the encoder hidden states is then calculated, illustrated by the blue box in Fig. 6. The block takes in the encoder hidden states, as well as the attention weights, and outputs a weighted sum.

The architecture of each decoder cell is illustrated in the red box in the upper right of Fig. 6. The input to each RNN cell is a concatenation of the previous prediction, $\hat{\mathbf{y}}_{t-1}$, with the weighted encoder hidden states, $\mathbf{w}$. Furthermore, the linear layer that conducts the prediction for each state, $\hat{\mathbf{y}}_t$, takes $\hat{\mathbf{y}}_{t-1}$, and $\mathbf{w}$ as input. Each prediction can, therefore, look at the entire past trajectory, and identify relevant parts to conduct as accurate a prediction as possible. The linear layer is allowed to look at the current input, to further enhance the accuracy of the prediction, where short-term dependencies can be directly determined. For the case of ship trajectory prediction, the next state will undoubtedly have a high dependency on the previous.

Each local model is, therefore, trained using the outlined architecture. The decoder will function in the same manner as in Section 2.3.2, where states are iteratively predicted, but instead of reconstructing the input, a target sequence, $\mathbf{y}$, is predicted. The loss is calculated using the mean squared error as in Section 2.3.2. To optimally train the network, all combinations of past and future 30 min trajectory segments should be utilized. In this study, one hour trajectory segments were extracted using a sliding window technique, where the window size was one minute. The first 30 min of each trajectory are defined as the input (i.e. past), and the final 30 the target (i.e. future).

3. Results and discussion

In this section, the results from a case study using the outlined deep learning framework are presented. A data set corresponding to one year of AIS data from January 1st 2017 to January 1st 2018 for the region around the city of Tromsø, Norway was utilized. This region contains complex traffic, and provided a relevant test case for the framework.

PyTorch [59] was utilized to implement the neural networks. All networks were trained using the Adam optimizer [61]. Furthermore, gradient clipping [62] and batch normalization [63] were utilized to aid in convergence. Hyperparameters were tuned for this specific region, and will need to be tuned to the specific geographical region to which the framework is to be applied.

3.1. Clustering module

In this section, the results for the clustering module are presented. The technique described in Section 2.3 was applied to the data set corresponding to the region surrounding Tromsø. This corresponded to approximately 70,000 trajectories. In this study, a VRAE was utilized to cluster the historical AIS trajectories. The results indicated that using a VRAE as opposed to a β-VRAE resulted in more optimal clusters for the architecture and data in this study, i.e. $\beta = 1$. It appeared based on visual inspection of the clusters of trajectories, that increasing the value of β caused multiple local behavior clusters to merge. This may degrade the results of the subsequent trajectory prediction, as it is desirable to discover behavior clusters that are as specific as possible. It should be noted that the optimal value of β will vary based on the complexity of the ship traffic in the region of interest.

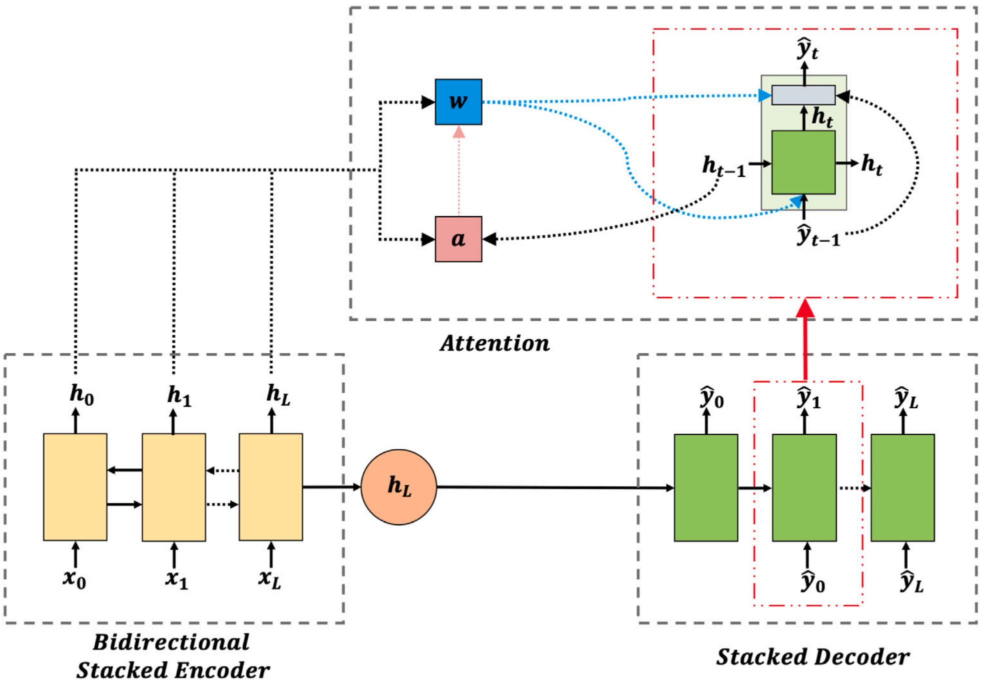

Fig. 6. Prediction architecture using a sequence-to-sequence model with attention. The bidirectional stacked encoder is illustrated in yellow, and the stacked decoder in green. For each prediction in the decoder, the attention mechanism looks at the encoder hidden states and calculates a weighted sum. This is then input to the decoder RNN cell.

3.1.1. Network training

The VRAE utilized in this study was comprised of a stacked, bidirectional GRU encoder with 3 layers, each with a hidden size of 50. A stacked GRU decoder with 2 layers, and a hidden size of 50 was used. The dimensionality of the latent space was set to 20 to allow for further compression of the data. A number of variations of parameters were run to determine the best performance.

During training, the data set was shuffled and split into training and validation data sets. The training data accounted for 90% of the trajectories, and the validation 10%. Fig. 7 illustrates the total loss of the VRAE on both the training and validation sets during training for 10 epochs. The results indicate that the model is not overfitting to the data, as the validation and training losses are highly correlated for the duration of the training. It appears that the total loss increases over time, but this effect is due to a technique known as KL-annealing, where the KL-loss term is introduced linearly over a span of a number of epochs. In this study, it was introduced over five epochs, as can be seen in Fig. 8. In this figure, the reconstruction- and KL-loss terms are plotted individually. KL-annealing allows the model to learn how to reconstruct the data before enforcing the KL-regularization term. As a result, it can be seen that the reconstruction loss decreased quickly, whilst the KL-term increased. Each step of the KL-loss downwards after this corresponded to an increase in its weighting. The loss terms converged after this, and it was concluded that the model had converged.

3.1.2. Clustering results

In order to cluster the trajectories, a forward pass of the encoder was run to generate the latent representations for each trajectory. The trajectories are then clustered in this space using HDBSCAN. The implementation in [64] was utilized in this study.

The minimum cluster size in the algorithm, however, is found to be decisive in the type of clusters discovered. This value was varied, and found to play a significant role in the outcome of the remainder of the architecture. When the minimum cluster size was set to 10, over 400 clusters of vessel behavior were discovered. In this case, the algorithm is able to discover very specific vessel behavior. This is beneficial as one wishes to discriminate between behavior clusters, and generate predictions using these clusters. More specific behavior should lead

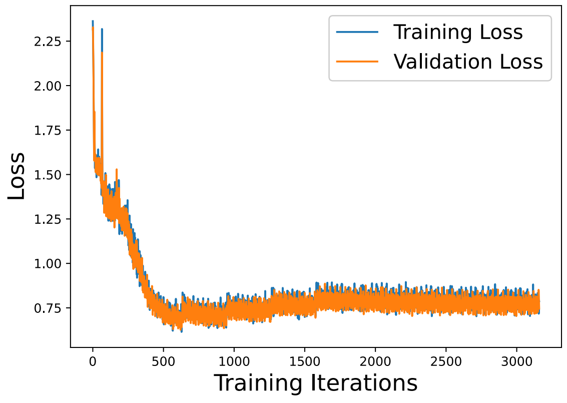

Fig. 7. VRAE loss.

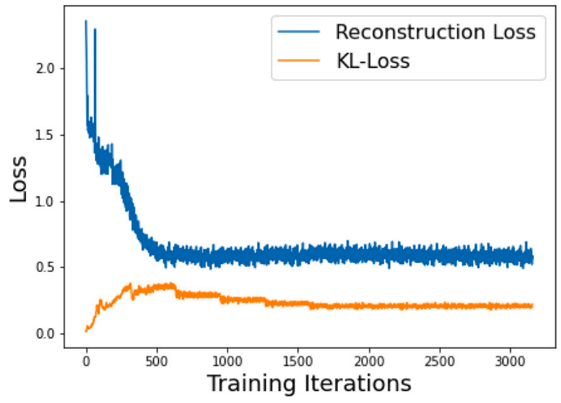

Fig. 8. Reconstruction and KL losses for VRAE.

to better predictions. However, when classifying a 30 min trajectory segment to one of these clusters in the classification module, the

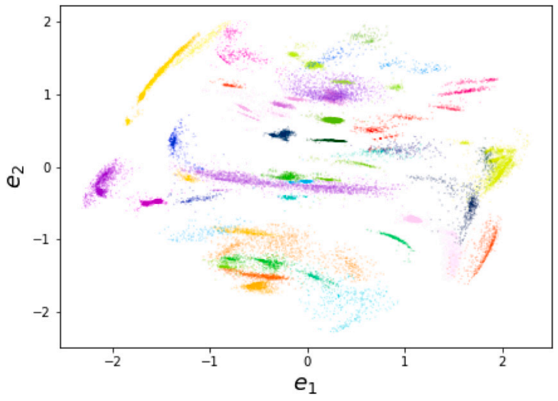

Fig. 9. Clusters in latent space of the VRAE. The latent space is illustrated using the top two principle components of the space, e_1 and e_2. Clusters with similar colors are not necessarily the same.

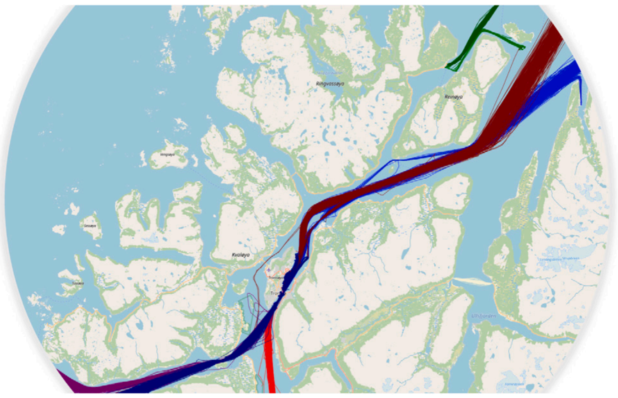

Fig. 10. Subset of discovered trajectory clusters with minimum cluster size of 100.

network will have great difficulty in correctly classifying the segment, and the performance of the overall algorithm will be degraded. This is due to the existence of many similar clusters, such that the algorithm is unable to conduct an accurate classification. Furthermore, such small clusters will not have sufficient data to train a prediction model.

Increasing the minimum cluster size causes clusters of local behavior to merge into larger clusters, where the behavior within a given cluster varies to a greater degree. Discovering smaller clusters allows the model to discover a greater number of ship speed clusters within a given route for instance. Merging these clusters, however, allows the model to classify a given trajectory segment with a higher degree of accuracy, contributing to the overall success of the algorithm. As a result, a minimum cluster size of 100 was set based on trials for various minimum cluster sizes.

However, clusters of only 100 trajectories may prove to be insufficient to train a prediction model in some cases. Further work should be conducted on the required size of a cluster to effectively train a model. Furthermore, by expanding the data set input to the framework, the size of clusters should grow, due to the increased occurrence of historical ship behavior that belongs to such smaller clusters. In general, increasing the amount of data utilized will aid the deep learning architectures outlined in this study. Nonetheless, the results of this study indicate the potential of the developed method to facilitate effective predictions. In the case of this study, however, most clusters were comprised thousands of trajectories which should provide enough data to train the relevant models. If implemented in a actual system, the user should also be made aware of models that are trained on limited data, if such data clusters exist.

In this case, 52 ship behavior clusters were discovered. Fig. 9 illustrates the clustered latent representations, where it appears that the algorithm had discovered meaningful clusters. Fig. 10 illustrates a subset of the clusters discovered by the module. Each of these clusters corresponds to a cluster of historical ship behavior. Despite utilizing clusters of more general behavior, discovering such main local ship behavior clusters will aid the performance of the local prediction module, which, with its architecture, can predict various behavior within these main clusters.

3.2. Classification module

The classification module was designed with a bidirectional, 5-layer stacked GRU with 20 hidden units as the encoder. 3 linear layers were utilized as the classification head, where the first was set to have one fourth as many neurons as the number of classes (i.e. number of clusters), the second half as many, and the third as many neurons as the number of classes. The embedding size was set to 10. During training,

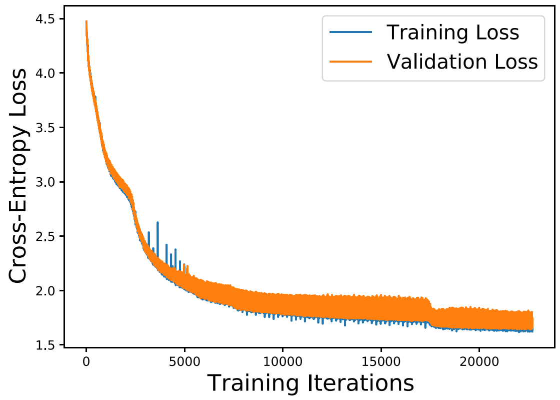

Fig. 11. Classification network loss, where the loss is defined as the cross-entropy loss.

Table 1
Size of data sets.

Training	Validation	Test
1.127×10^6	1.66×10^5	3.16×10^5

it was found that embedding the ship type led to the model focusing too much on the ship type, thereby degrading the results. As a result, a dropout rate of 50% [65] was applied to the embedding layer to prevent overfitting to the embedding data.

3.2.1. Network training

Prior to training the network, the data set was shuffled and split into training (70%), validation (10%) and test (20%) sets. Subsequently, each data set was split into 30 min trajectory segments using a sliding window technique with a window size of one minute. As a result, all possible 30 min trajectory segments in the data are used to train the classifier. The size of the data sets is shown in Table 1.

The results of the training are illustrated in Fig. 11. It appears that both the training and validation losses continue to decrease until about 20000 iterations. As a result, it was concluded that the model had converged at this point. Furthermore, the validation loss is closely correlated with the training loss. Therefore, it was concluded that the model was not overfitting to the data.

3.2.2. Classification results

When evaluating the classification performance on the test set, the classification accuracy was found to be 47%. However, in this study it is suggested to use the softmax distribution of possible clusters. Any clusters with a softmax output over 0.1 (i.e. 10% probability) are output as likely ship behavior clusters. In this manner, the model

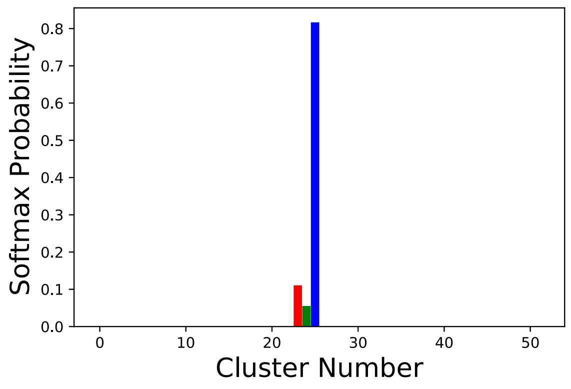

Fig. 12. Softmax probability distribution for selected vessel case, illustrating uncertainty of cluster assignment.

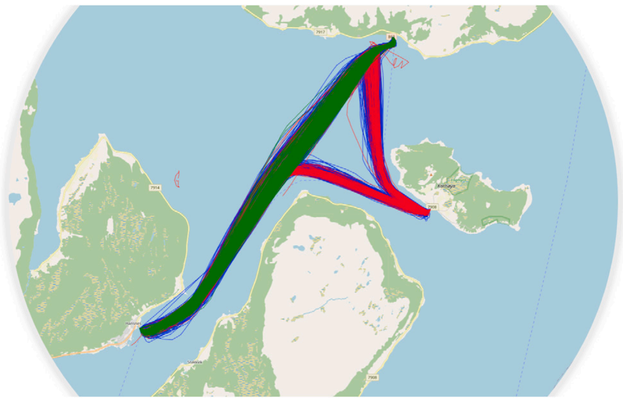

Fig. 13. Classified ship behavior clusters for selected vessel case. The selected vessel was classified to the blue, with the red above the softmax threshold of 10%, and the green above 5%.

Table 2
Classification accuracy and average number of clusters.

	Absolute	10% Softmax	5% Softmax
Accuracy	47%	73%	92%
Clusters	–	3	5

identifies multiple possible clusters the trajectory segment may belong to. The softmax distribution for a randomly selected vessel trajectory segment from the test set is illustrated in Fig. 12. The colors of the bars correspond to the colors of the trajectory clusters illustrated in Fig. 13.

For the selected vessel in Fig. 12, the model correctly classified the behavior to cluster 25 (i.e. blue). However, the softmax output also indicated that the selected vessel may belong to cluster 23 (i.e. red), as its probability was above 0.1. Furthermore, cluster 24 had a probability above 0.05, and might have been a possible behavior cluster. All three clusters share common behavior, and in this case, it appeared appropriate to investigate multiple possible clusters.

When using a 10% softmax threshold, the classification accuracy increased to 73% for the test set. On average, the model outputs 3 possible clusters a trajectory segment may belong to. Decreasing the threshold was also investigated. With a threshold of 5%, the accuracy rate increased to 92%. In this case, the model outputs an average of 5 possible clusters the trajectory segment may belong to. The results are summarized in Table 2. For the selected vessel case in Figs. 12 and 13, the blue and red clusters would be identified for a softmax threshold of 10%, with the addition of the green cluster for a threshold of 5%.

Table 3
Size of local behavior model data sets.

	Training	Validation	Test
Model 1	9.23×10^4	2.80×10^4	1.2×10^4
Model 2	1.02×10^4	3.3×10^3	1.3×10^3

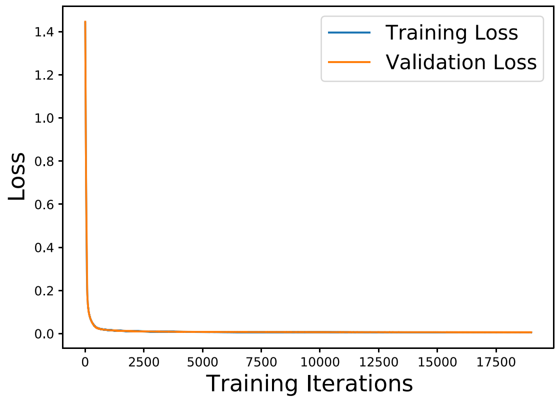

Fig. 14. Model 1 training and validation loss.

Overall, it appears that the model was successful in classifying the trajectory segments in the test set, where the accuracy increased as the softmax probability threshold was lowered.

3.3. Local behavior module

In the local behavior module, local models for each cluster of ship behavior are available. In this study, however, it was infeasible to train 52 neural networks to evaluate the overall performance using the available resources. In a commercial setting, however, this should be done. As such, only two models were trained to illustrate the performance of the method. These correspond to the blue and red clusters from the example in Section 3.2.2, illustrated in Fig. 13. These models were chosen as they were above the 10% softmax threshold used in this study. These models are hereafter referred to as model 1 (i.e. the blue cluster), and model 2 (i.e. the red cluster).

Both models have a bidirectional 2-layer stacked GRU encoder with 20 hidden units, and a 2-layer stacked GRU decoder with 20 hidden units. Variations of these architectures were evaluated to determine the architecture with the best performance.

3.3.1. Network training

The data sets for both models were initially reduced to only contain the trajectories in the respective clusters. Subsequently, each model data set was shuffled and split into training (70%), validation (10%) and test (20%) sets. These data sets were again split into 30 min trajectory segments using a sliding window technique with a window size of one minute. Source sequences (past 30 min trajectory), and their corresponding target sequences (future 30 min trajectory) were extracted in this manner. The sizes of the respective data sets are summarized in Table 3.

The training and validation losses for model 1 and model 2 are illustrated in Figs. 14 and 15, respectively. Both models were trained for 1000 epochs. The training and validation losses were both correlated for the duration the training of the models, indicating that neither model overfit to the data. The losses continue to decrease for the duration of the training iterations illustrated in the figures. Once the decrease was minimal, the models were assumed to have converged.

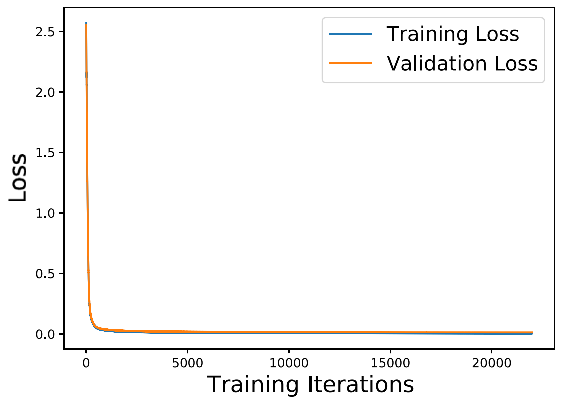

Fig. 15. Model 2 training and validation loss.

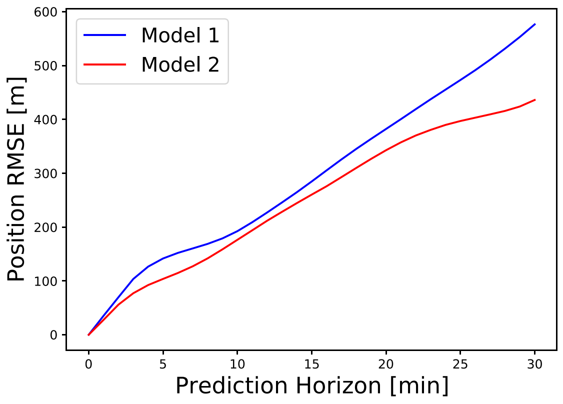

Fig. 16. Root mean squared error (RMSE) of test set predictions.

3.3.2. Prediction results

The predictive performance of the models was evaluated on their respective test sets. The results for each model are presented in Fig. 16. The figure illustrates the root mean squared error (RMSE) of the predicted position as a function of the prediction horizon. The results indicate that model 2 has better performance, with a mean squared error of 436 m for a prediction horizon of 30 min, whilst model 1 has a mean squared error of 576 m. This may be due to model 1 being trained on a greater number of trajectories. The cluster for model 1 was much larger than for model 2. As a result, there will be a greater degree of variation in the data, and the model is not as effective in capturing the variation. In model 2, the data likely has less variation, and, therefore, fits well to the data.

Fig. 17 illustrates the prediction results for the selected vessel case in Section 3.2.2. Here, two predictions were conducted using the two models identified by the classification module. The results indicate the sequence-to-sequence model with attention can provide successful results, even with highly nonlinear input trajectories. It appears that the attention mechanism allows the models to focus on the most relevant aspects of the past trajectory. Model 2, however, appears to have better performance than model 1. Model 1 should have better performance, as the test trajectory belongs to the cluster for model 1. Nonetheless, in this case it appears that the model with the second highest probability has the best performance. This indicates that incorrectly classifying the selected vessel may not result in a significant degradation in the prediction results. As such, many of the incorrectly classified trajectory segments may be classified to a cluster of similar behavior.

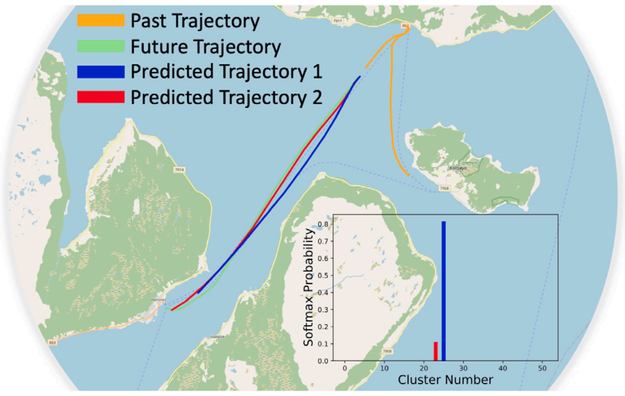

Fig. 17. Predictions for selected vessel case.

3.4. Framework performance

In this section, the performance of the overall framework is evaluated relative to a baseline model. The most relevant work for comparison is arguably that presented in [40], where sequence-to-sequence RNNs were utilized to predict ship trajectories using historical AIS data. Forti et al. [40] showed the superior performance of using sequence-to-sequence models compared to an Ornstein–Uhlenbeck stochastic process applied in similar studies.

Seeing as sequence-to-sequence models provide the basis for many of the functions in this framework, it was concluded that it would provide an ideal baseline for comparison. The framework in this study leverages the ability to identify clusters of specific historical ship behavior, upon which local behavior models can be trained to facilitate enhanced predictions. To support this argument, such local models facilitated via the framework in this study are compared to global models that are trained on all available data in the region.

The global model is comprised of a sequence-to-sequence model, as in [40], and is trained on all trajectories in the region. By using the framework in this study, a local model is also trained on the data in a specific cluster using the same sequence-to-sequence architecture. To evaluate the performance, the models were applied to the data in cluster 25 in Section 3.2.2.

Fig. 18 illustrates the RMSE of the predicted position by applying various models. The dashed orange line indicates the results from training a global model on all trajectories in the region, and the results of the local model via the dashed green line. It is evident that utilizing local models via the outlined framework in this study results in enhanced predictive performance compared to the baseline sequence-to-sequence model, i.e. [40].

Furthermore, the local behavior models in this study apply an attention mechanism to enhance the predictions. The effect of this mechanism was, therefore, also investigated. The solid orange line illustrates the results of a global model with attention applied to the same cluster, where it is evident that the attention mechanism improves the predictive performance. Similarly, a local model with attention has superior performance, as indicated by the solid green line. The local model with attention corresponds to the technique suggested in this study, where the results indicate that it is capable of achieving the most accurate results. Seeing as the framework is designed to support proactive collision avoidance actions, the model suggested in this study should best support this goal.

4. Conclusion and further work

Limited work has been conducted on utilizing deep learning to enhance the safety of maritime transportation systems. One area of interest is in aiding maritime situation awareness via proactive collision

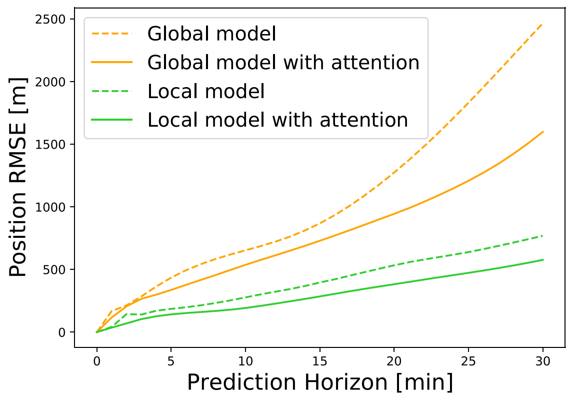

Fig. 18. Comparison of sequence-to-sequence model performance.

avoidance. To facilitate this, global scale trajectory predictions, i.e. order 5–30 min, should be conducted. This study suggests an approach to this issue via a deep learning framework for regional trajectory prediction. Using such an architecture, the future trajectory of a vessel can be predicted in under a second. The framework investigates utilizing modern deep learning architectures to facilitate a decomposition of regional ship behavior into local models. Utilizing historical AIS data, the framework is successful in clustering historical ship behavior using a variational recurrent autoencoder. The results also indicate that the increased complexity of the model allows it to cluster vessel behavior more successfully.

Utilizing this historical knowledge, the past behavior of a selected vessel is classified to the most likely clusters of historical behavior. Predictions corresponding to the behavior in each cluster are then output to the user. The local prediction models are comprised of sequence-to-sequence models with attention. The results indicate that the attention mechanism assists the prediction by allowing the model to focus on the most relevant parts of the past trajectory. By decomposing the behavior into local models, greater accuracy can be achieved than training a similar prediction model on the data in all clusters. Overall, the suggested framework is successful in predicting trajectories on a global scale.

Further work will include providing further uncertainty estimation via Bayesian dropout techniques. In this manner, a distribution is predicted for each time step. The classification module will also be further improved to enhance the classification accuracy. Weather parameters will likely aid the predictions, as ships will display various behavior based on the prevailing weather conditions. This will also be addressed in future work. Finally, methods to automatically tune hyperparameters in the networks will be investigated.

CRediT authorship contribution statement

Brian Murray: Conceptualization, Methodology, Software, Formal analysis, Writing - original draft, Visualization. **Lokukaluge Prasad Perera:** Conceptualization, Writing - review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the Norwegian Ministry of Education and Research, Norway and the MARKOM-2020 project, Norway, a development project for maritime competence established by the Norwegian Ministry of Education and Research in cooperation with the Norwegian Ministry of Trade, Industry and Fisheries. The authors would also like to express gratitude to the Norwegian Coastal Administration for providing access to their AIS database. Map data copyrighted OpenStreetMap contributors and available from https://www.openstreetmap.org.

References

[1] Montewka J, Ehlers S, Goerlandt F, Hinz T, Tabri K, Kujala P. A framework for risk assessment for maritime transportation systems - A case study for open sea collisions involving RoPax vessels. Reliab Eng Syst Saf 2014;124:142–57. http://dx.doi.org/10.1016/j.ress.2013.11.014.
[2] Goerlandt F, Montewka J. Maritime transportation risk analysis: Review and analysis in light of some foundational issues. Reliab Eng Syst Saf 2015;138:115–34. http://dx.doi.org/10.1016/j.ress.2015.01.025.
[3] Dinis D, Teixeira AP, Guedes Soares C. Probabilistic approach for characterising the static risk of ships using Bayesian networks. Reliab Eng Syst Saf 2020;203:107073. http://dx.doi.org/10.1016/j.ress.2020.107073.
[4] Endsley MR, Bolté B, Jones DG. Designing for situation awareness: An approach to user-centered design. Taylor & Francis; 2003.
[5] Perera LP, Guedes Soares C. Collision risk detection and quantification in ship navigation with integrated bridge systems. Ocean Eng 2015;109:344–54. http://dx.doi.org/10.1016/j.oceaneng.2015.08.016.
[6] Perera LP, Carvalho JP, Guedes Soares C. Autonomous guidance and navigation based on the COLREGs rules and regulations of collision avoidance. Adv Ship Design Pollut Prev 2010;205–16. http://dx.doi.org/10.1201/b10565-26.
[7] Yu Q, Liu K, Yang Z, Wang H, Yang Z. Geometrical risk evaluation of the collisions between ships and offshore installations using rule-based Bayesian reasoning. Reliab Eng Syst Saf 2021;210:107474. http://dx.doi.org/10.1016/j.ress.2021.107474.
[8] Xiao Z, Fu X, Zhang L, Goh RSM. Traffic pattern mining and forecasting technologies in maritime traffic service networks: A comprehensive survey. IEEE Trans Intell Transp Syst 2020;21(5):1796–825. http://dx.doi.org/10.1109/TITS.2019.2908191.
[9] Perera LP, Oliveira P, Guedes Soares C. Maritime traffic monitoring based on vessel detection, tracking, state estimation, and trajectory prediction. IEEE Trans Intell Transp Syst 2012;13(3):1188–200. http://dx.doi.org/10.1109/TITS.2012.2187282.
[10] Perera LP. Navigation vector based ship maneuvering prediction. Ocean Eng 2017;138:151–60. http://dx.doi.org/10.1016/j.oceaneng.2017.04.017.
[11] Perera LP, Murray B. Situation awareness of autonomous ship navigation in a mixed environment under advanced ship predictor. In: Proceedings of the international conference on offshore mechanics and arctic engineering, 7B-2019, American Society of Mechanical Engineers (ASME); 2019, http://dx.doi.org/10.1115/OMAE2019-95571.
[12] Daranda A. Neural network approach to predict marine traffic. Balt J Modern Comput 2016;4(3):483–95.
[13] Tu E, Zhang G, Rachmawati L, Rajabally E, Huang G-B. Exploiting AIS data for intelligent maritime navigation: A comprehensive survey from data to methodology. IEEE Trans Intell Transp Syst 2017;1–24. http://dx.doi.org/10.1109/TITS.2017.2724551.
[14] Montewka J, Hinz T, Kujala P, Matusiak J. Probability modelling of vessel collisions. Reliab Eng Syst Saf 2010;95(5):573–89. http://dx.doi.org/10.1016/J.RESS.2010.01.009.
[15] Goerlandt F, Kujala P. Traffic simulation based ship collision probability modeling. In: Reliability engineering and system safety. Elsevier; 2011, p. 91–107. http://dx.doi.org/10.1016/j.ress.2010.09.003.
[16] Bye RJ, Aalberg AL. Maritime navigation accidents and risk indicators: An exploratory statistical analysis using AIS data and accident reports. Reliab Eng Syst Saf 2018;176:174–86. http://dx.doi.org/10.1016/j.ress.2018.03.033.
[17] Silveira PAM, Teixeira AP, Soares CG. Use of AIS data to characterise marine traffic patterns and ship collision risk off the coast of portugal. J Navig 2013;66(06):879–98. http://dx.doi.org/10.1017/S0373463313000519, URL: http://www.journals.cambridge.org/abstract_S0373463313000519.
[18] Rong H, Teixeira AP, Guedes Soares C. Data mining approach to shipping route characterization and anomaly detection based on AIS data. Ocean Eng 2020;198. http://dx.doi.org/10.1016/j.oceaneng.2020.106936.
[19] Yu Q, Liu K, Chang CH, Yang Z. Realising advanced risk assessment of vessel traffic flows near offshore wind farms. Reliab Eng Syst Saf 2020;203:107086. http://dx.doi.org/10.1016/j.ress.2020.107086.
[20] Du L, Goerlandt F, Kujala P. Review and analysis of methods for assessing maritime waterway risk based on non-accident critical events detected from AIS data. Reliab Eng Syst Saf 2020;200:106933. http://dx.doi.org/10.1016/j.ress.2020.106933.
[21] Ristic B, Scala BL, Morelande M, Gordon N. Statistical analysis of motion patterns in AIS data: Anomaly detection and motion prediction. In: 2008 11th International Conference on Information Fusion. 2008, p. 40–6. http://dx.doi.org/10.1109/ICIF.2008.4632190.

[22] Pallotta G, Vespe M, Bryan K. Vessel pattern knowledge discovery from AIS data: A framework for anomaly detection and route prediction. Entropy 2013;15(12):2218–45. http://dx.doi.org/10.3390/e15062218.
[23] Pallotta G, Horn S, Braca P, Bryan K. Context-Enhanced Vessel Prediction Based On Ornstein-Uhlenbeck Processes Using Historical AIS Traffic Patterns: Real-World Experimental Results. In: Information fusion, 2014 17th international conference on. 2014. p. 1–7.
[24] Mazzarella F, Arguedas VF, Vespe M. Knowledge-based vessel position prediction using historical AIS data. In: 2015 sensor data fusion: trends, solutions, applications. IEEE; 2015, p. 1–6. http://dx.doi.org/10.1109/SDF.2015.7347707.
[25] Xiao Z, Ponnambalam L, Fu X, Zhang W. Maritime traffic probabilistic forecasting based on vessels' waterway patterns and motion behaviors. IEEE Trans Intell Transp Syst 2017;18(11):3122–34. http://dx.doi.org/10.1109/TITS.2017.2681810.
[26] Hexeberg S, Flaten AL, Eriksen B-OH, Brekke EF. AIS-Based vessel trajectory prediction. In: 2017 20th international conference on information fusion. IEEE; 2017, http://dx.doi.org/10.23919/ICIF.2017.8009762.
[27] Dalsnes BR, Hexeberg S, Flåten AL, Eriksen B-OH, Brekke EF. The neighbor course distribution method with Gaussian mixture models for AIS-based vessel trajectory prediction. In: 2018 21st international conference on information fusion. IEEE; 2018, p. 580–7.
[28] Rong H, Teixeira AP, Guedes Soares C. Ship trajectory uncertainty prediction based on a Gaussian process model. Ocean Eng 2019;182:499–511. http://dx.doi.org/10.1016/J.OCEANENG.2019.04.024.
[29] Murray B, Perera LP. A dual linear autoencoder approach for vessel trajectory prediction using historical AIS data. Ocean Eng 2020;209:107478. http://dx.doi.org/10.1016/j.oceaneng.2020.107478.
[30] Xu Z, Saleh JH. Machine learning for reliability engineering and safety applications: Review of current status and future opportunities. Reliab Eng Syst Saf 2021;107530. http://dx.doi.org/10.1016/j.ress.2021.107530.
[31] Zhang Y, Sun X, Chen J, Cheng C. Spatial patterns and characteristics of global maritime accidents. Reliab Eng Syst Saf 2021;206:107310. http://dx.doi.org/10.1016/j.ress.2020.107310.
[32] Goodfellow I, Bengio Y, Courville A. Deep learning. MIT Press; 2016.
[33] Nguyen KTP, Medjaher K. A new dynamic predictive maintenance framework using deep learning for failure prognostics. Reliab Eng Syst Saf 2019;188:251–62. http://dx.doi.org/10.1016/j.ress.2019.03.018.
[34] Zhang W, Feng X, Goerlandt F, Liu Q. Towards a convolutional neural network model for classifying regional ship collision risk levels for waterway risk analysis. Reliab Eng Syst Saf 2020;204:107127. http://dx.doi.org/10.1016/j.ress.2020.107127.
[35] Nguyen D, Vadaine R, Hajduch G, Garello R, Fablet R. 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, 2018, p. 331–40.
[36] Nguyen D, Vadaine R, Hajduch G, Garello R, Fablet R. GeoTrackNet–A Maritime anomaly detector using probabilistic neural network representation of AIS tracks and a contrario detection. IEEE Trans Intell Transp Syst 2021;1–13. http://dx.doi.org/10.1109/TITS.2021.3055614.
[37] Zhang W, Li X, Ma H, Luo Z, Li X. Transfer learning using deep representation regularization in remaining useful life prediction across operating conditions. Reliab Eng Syst Saf 2021;211:107556. http://dx.doi.org/10.1016/j.ress.2021.107556.
[38] Yao D, Zhang C, Zhu Z, Huang J, Bi J. Trajectory clustering via deep representation learning. In: Proceedings of the international joint conference on neural networks, 2017-May, Institute of Electrical and Electronics Engineers Inc.; 2017, p. 3880–7. http://dx.doi.org/10.1109/IJCNN.2017.7966345.
[39] Murray B, Perera LP. Deep representation learning-based vessel trajectory clustering for situation awareness in ship navigation. In: Maritime technology and engineering 5. Proceedings of the 5th international conference on maritime technology and engineering (MARTECH 2020), 2021, p. 157-166, http://dx.doi.org/10.1201/9781003171072.
[40] Forti N, Millefiori LM, Braca P, Willett P. Prediction of vessel trajectories from AIS data via sequence-to-sequence recurrent neural networks. In: IEEE international conference on acoustics, speech and signal processing - proceedings. 2020-May, Institute of Electrical and Electronics Engineers Inc.; 2020, p. 8936–40. http://dx.doi.org/10.1109/ICASSP40776.2020.9054421.
[41] Capobianco S, Millefiori LM, Forti N, Braca P, Willett P. Deep learning methods for vessel trajectory prediction based on recurrent neural networks. 2021, URL: http://arxiv.org/abs/2101.02486.
[42] Wang J, Zhang C. Software reliability prediction using a deep learning model based on the RNN encoder–decoder. Reliab Eng Syst Saf 2018;170:73–82. http://dx.doi.org/10.1016/j.ress.2017.10.019.
[43] Utne IB, Rokseth B, Sørensen AJ, Vinnem JE. Towards supervisory risk control of autonomous ships. Reliab Eng Syst Saf 2020;196:106757. http://dx.doi.org/10.1016/j.ress.2019.106757.
[44] Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Nature 1986;323(6088):533–6. http://dx.doi.org/10.1038/323533a0.
[45] Bengio Y, Simard P, Frasconi P. Learning long-term dependencies with gradient descent is difficult. IEEE Trans Neural Netw 1994;5(2):157–66. http://dx.doi.org/10.1109/72.279181.
[46] Cho K, van Merrienboer B, Bahdanau D, Bengio Y. On the properties of neural machine translation: Encoder-decoder approaches. Association for Computational Linguistics (ACL); 2014, URL: arXiv:1409.1259.
[47] Chung J, Gulcehre C, Cho K, Bengio Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. In: NIPS'2014 deep learning workshop. 2014, URL: arXiv:1412.3555.
[48] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput 1997;9(8):1735–80. http://dx.doi.org/10.1162/neco.1997.9.8.1735.
[49] Schuster M, Paliwal KK. Bidirectional recurrent neural networks. IEEE Trans Signal Process 1997;45(11):2673–81. http://dx.doi.org/10.1109/78.650093.
[50] Graves A, Mohamed AR, Hinton G. Speech recognition with deep recurrent neural networks. In: IEEE international conference on acoustics, speech and signal processing - proceedings. 2013, p. 6645–9. http://dx.doi.org/10.1109/ICASSP.2013.6638947, URL: arXiv:1303.5778.
[51] Higgins I, Matthey L, Pal A, Burgess C, Glorot X, Botvinick MM, et al. beta-VAE: Learning basic visual concepts with a constrained variational framework, In: Proceedings of the international conference on learning representations, 2017.
[52] Campello RJGB, Moulavi D, Sander J. Density-based clustering based on hierarchical density estimates. In: Pei J, Tseng VS, Cao L, Motoda H, Xu G, editors. Advances in knowledge discovery and data mining. Berlin, Heidelberg: Springer Berlin Heidelberg; 2013, p. 160–72.
[53] Srivastava N, Mansimov E, Salakhutdinov R. Unsupervised learning of video representations using LSTMs. In: 32nd international conference on machine learning, 1, International Machine Learning Society (IMLS); 2015, p. 843–52, URL: arXiv:1502.04681.
[54] Sutskever I, Vinyals O, Le QV. Sequence to sequence learning with neural networks. Adv Neural Inf Process Syst 2014;4:3104–12, URL: arXiv:1409.3215.
[55] Kingma DP, Welling M. Auto-encoding variational Bayes. In: Proceedings of the international conference on learning representations. 2014, URL: arXiv:1312.6114.
[56] Rezende DJ, Mohamed S, Wierstra D. Stochastic backpropagation and approximate inference in deep generative models. In: Proceedings of the 31st international conference on international conference on machine learning, vol. 32, 2014.
[57] Fabius O, van Amersfoort JR. Variational recurrent auto-encoders. In: Proceedings of the international conference on learning representations. 2015, URL: http://arxiv.org/abs/1412.6581.
[58] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. In: 1st international conference on learning representations - workshop track proceedings. International Conference on Learning Representations, ICLR; 2013, URL: arXiv:1301.3781.
[59] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. PyTorch: An imperative style, high-performance deep learning library. In: Wallach H, Larochelle H, Beygelzimer A, d' Alché-Buc F, Fox E, Garnett R, editors. Advances in neural information processing systems 32. Curran Associates, Inc.; 2019, p. 8024–35, URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.
[60] Bahdanau D, Cho KH, Bengio Y. Neural machine translation by jointly learning to align and translate. In: 3rd international conference on learning representations, ICLR 2015 - Conference track proceedings. 2015, URL: arXiv:1409.0473.
[61] Kingma DP, Ba JL. Adam: A method for stochastic optimization. In: Proceedings of the international conference on learning representations. 2015, URL: arXiv:1412.6980.
[62] Pascanu R, Mikolov T, Bengio Y. On the difficulty of training recurrent neural networks. In: 30th international conference on machine learning. International Machine Learning Society (IMLS); 2013, p. 2347–55, URL: arXiv:1211.5063.
[63] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: 32nd international conference on machine learning, vol. 1. International Machine Learning Society (IMLS); 2015, p. 448–56, URL: arXiv:1502.03167.
[64] McInnes L, Healy J, Astels S. hdbscan: Hierarchical density based clustering. J Open Source Softw 2017;2(11):205.
[65] Srivastava N, Hinton G, Krizhevsky A, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 2014;15:1929–58. http://dx.doi.org/10.5555/2627435.2670313.