Predicting the Time of Arrival at Port for Maritime Surface Ships in the Baltic Sea Using Recurrent Neural Networks

Jonatan Lahtivuori
40601-308-2015
Supervisors: Sebastien Lafond &
Sepinoud Azimi Rashti

Abstract

Here is where I would say what is in this document.

Contents

1 Introduction

Describe the need for accurate ETA predictions for maritime traffic and what data is available what has been done already and what the thesis will contribute.

Structure of the thesis.

2 Maritime vessels and traffic

The Baltic Sea, with an approximate surface area of 420,000 km^2, is rather small compared to other seas and navigating in the Baltic Sea is often confined to fairways, especially when coming closer to land and travelling in the archipelago. This creates highways of vessel traffic coming from and going to the many ports in the Baltic Sea merging into one another at some point during the voyage. This is even more evident in the winter months when there are restrictions for where vessels are allowed to navigate due to the ice, which is governed by the Finnish and Swedish authorities [1]. During the winter months, vessels might also be required to be assisted by an icebreaker to enter the port. With only around 30 operational icebreakers in the Baltic Sea, queues could even be created for assistance where vessels have to anchor until assistance arrives. Other characteristics are the areas of international waters in the Baltic Sea; vessels travelling from and to the eastern part of the Baltic Sea often use these areas as an example. To enter the Baltic Sea from the Atlantic, all vessels have to travel through the Danish straits and Kattegat.

In recent years, approximately 90% of all import and export of Finland was done by sea [2]. There is no question about the importance of sea transports and maintaining efficient operations at sea and at port is of interest for all parties, especially Finland where most of all import and export passes through ports at some point during the transportation.

One important port for the Finnish industry and economy is the Port of Naantali. Located close the city of Turku and having a central location in the Baltic Sea, it was deemed to be a suitable candidate for this thesis.

2.1 Automatic identification system

Automatic identification system (*AIS*) was introduced by the International Maritime Organization (*IMO*) in the early 2000's in accordance with the Safety of Life at Sea (*SOLAS*) treaty as an open communication tool for all maritime traffic. The main objectives of the treaty are to improve the safety of life at sea, protection of the marine environment and safety and efficiency of navigation [3].

AIS was developed as a tool for exchanging information vessel-to-vessel and vessel to base-station, to improve the situational awareness and as a system to track vessels. It was first used by vessel traffic services (*VTS*), but has since its introduction been exploited for many other services and studies. Some of these are collision avoidance [4], accidents during wintertime [1], the impact of ice on vessel performance [5], predicting vessels arrival time [6, 7, 8, 9, 10] and vessel tracking services such as MarineTraffic (`https://www.marinetraffic.com/`).

In its general operating form, AIS operates on two VHF channels and all vessels

or base stations within the vessels transmission range receive the messages transmitted and, vice versa, the vessel receives all messages broadcasted in its receiving range. The broadcasting is based on Self-organized Time Division Multiple Access (*(S)TDMA*) with a minimum of 2000 time slots per minute broadcasting capacity rate and the ship-to-ship communication for vessels closer to each other takes precedence over vessels farther away from one another, which allows for sharing time slots and, thus, overloading the available time slots [3].

In accordance with regulation V/19 of SOLAS, the IMO requires that all vessels with a gross tonnage of 300 or more on international voyages, cargo vessels of 500 gross tonnage or more not on international voyages and all passenger vessels disregarding the gross tonnage to be equipped with an AIS. Further, the EU requires new-built fishing vessels longer than 15 meters to be fitted with an AIS from November 2010 and existing vessels to install an AIS by May 2014 at the latest [11]. There are two types of AIS, Class A and Class B. Class A transceivers are more common and are compliant with the IMO regulations, whereas Class B transceivers are not subject to the full IMO AIS requirements. Class B transceivers are typically simpler, of lower cost and installed on smaller vessels of one's own choosing for improving the situational awareness. Class A transceivers take precedence over Class B transceivers when broadcasting.

The vessels onboard sensors and positioning systems interface with the AIS to allow for communicating the status of the vessel, both static and dynamic information. The AIS also allows for voyage-related information and safety-related information to be transmitted via the system, see Table 1. The AIS guidelines require that there is a minimum display and keyboard for input and receiving data, for updating information manually and retrieving data received via the AIS. Some data for the AIS are entered only once, some have to be entered for every voyage and others will be automatically gathered from the vessel's onboard sensors.

The information received by the AIS does not provide the full situational picture of the vessel's surroundings and should only be used as a navigational and situational awareness aid.

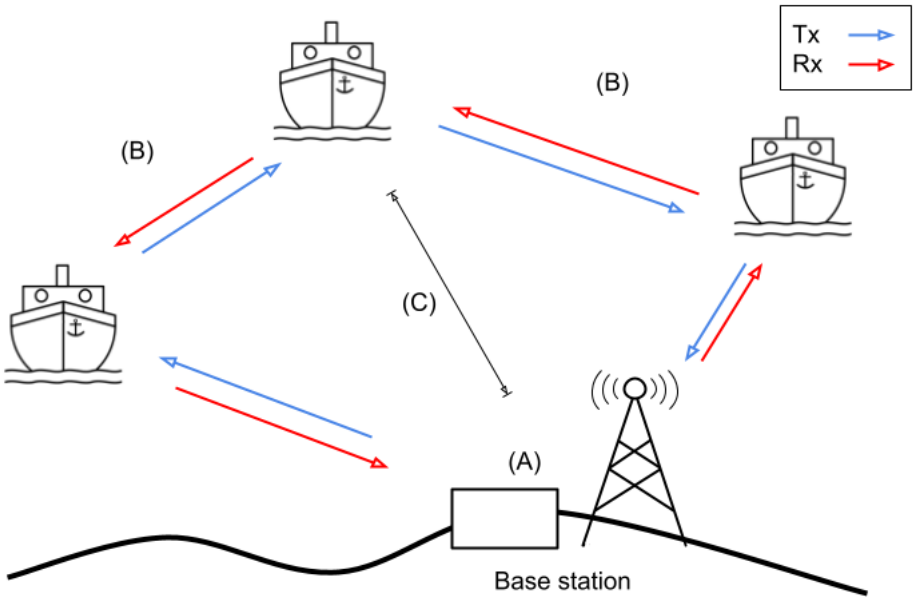

Figure 1: (A) Base station transmits information about other vessels, port data and possible hazards. Base station receives information from all vessels in range. (B) Ship-to-ship communication. Vessel broadcasts its current state (identifier, speed over ground, course over ground etc.) and receives data from all vessels in range, only limited by the number of possible time slots available. (C) Vessel outside the range of broadcasting AIS data to the base station, still broadcasts information to other vessels.

The AIS has a built-in integrity test (*BIIT*) that can automatically check that the AIS is operating without failures or malfunctions. If any failures or malfunctions are detected, an alarm is given to notify the user that the AIS is not operating as it should. The BIIT runs integrity tests continuously or at an interval deemed appropriate for the vessel's current situation. However, the BIIT does not validate the quality or accuracy of the data received from the onboard sensors of the vessel nor the data entered manually. This can introduce faulty data being broadcasted to surrounding vessels and could create dangerous situations. It is recommended that regular checks of the data transmitted by the AIS are validated for accuracy and quality [3].

AIS data is not encrypted nor authenticated when transmitted to other vessels or base stations. This, with the fact there is no validation of accuracy or quality, leads to poor veracity of the data transmitted. The information available through the AIS should not be trusted by itself and it is up to the OOW to safely operate the vessel and ensure safe navigation for the vessel and vessels in its surroundings. A study by Felski and Jaskolski found that the integrity of AIS data was for the most part reliable, but can not be the only trusted source of information regarding

other vessels intentions. The most useful information gained from the AIS was other vessels name, so that it would be possible to contact the vessel over radio and communicate with the vessel what its intentions are. The study also concluded that the veracity of dynamic data was better at open sea compared to in port areas where the speed was lower. As the majority of the duration of the routes will be at open sea moving at higher speed the veracity should be of minor concern for the work of this thesis.

The information available to transmit over AIS is divided into three groups, static, dynamic and voyage-related.

Data	Description
Static	
Maritime Mobile Service Identity (MMSI)	Set on installation, can change during vessel's operational lifespan
Call sign and name	Set on installation, can change during vessel's operational lifespan
IMO number	Set on installation
Length and beam	Set on installation or if changed
Type of ship	Selected from list of predefined values
Location of electronic positioning system	Set on installation, can be changed
Dynamic	
Vessel's position with accuracy indication and integrity status	Automatically updated from the position sensor
Position time stamp in UTC	Automatically updated from the main position sensor
Course over ground (COG)	Automatically updated from the main position sensor, if available
Speed over ground (SOG)	Automatically updated from the main position sensor
Heading	Automatically updated from the vessel's heading sensor
Navigational status	Manually entered by the OOW, as needed
Rate of turn (ROT)	Automatically updated from the vessel's ROT sensor or from the vessel's gyro
Voyage-related	
Draught	Manually entered at the start of the voyage
Hazardous cargo (type)	Manually entered at the start of the voyage confirming the presence of such cargo
Destination and ETA	Manually entered at the start of the voyage and updates, as needed
Route plan (waypoints)	Manually entered at the start of the voyage
Safety-related	
Short safety-related messages	Free format manually entered broadcasted to specific receiver or all vessels

Table 1: AIS data content and description, source [3].

Depending on the type of message sent by the AIS, the rate at which the AIS transmits messages autonomously changes. For static and voyage-related data the interval is six minutes between messages or at the request of another user. Dynamic information, in contrast, takes the vessel's current navigational status into consideration at which rate to transmit messages.

The different states of a vessel and at which rate it will transmit autonomously AIS messages with dynamic information can be seen in Table 2. Faster moving

vessels will broadcast their location and trajectory more frequently as a faster moving vessel will cover longer distances and can alter its course more over a shorter time. It is still important for the OOW to understand that not all vessels will be transmitting AIS data and, therefore, the AIS is not guaranteed to provide a complete overview of the surroundings. It is also not guaranteed that other vessels are receiving AIS data and could, therefore, be a risk factor.

Vessel state	**General reporting interval**
At anchor or moored and not moving faster than 3 knots	3 min
At anchor or moored and moving faster than 3 knots	10 s
0-14 knots	10 s
Changing course at 0-14 knots	3 1/3 s
14-23 knots	6 s
Changing course at 14-23 knots	2 s
Faster than 23 knots	2 s
Changing course at more than 23 knots	2 s

Table 2: AIS message broadcast interval for dynamic information for class A transceivers [3].

2.1.1 Estimated Time of Arrival

Estimated time of arrival is the estimated time when a vessel will reach its destination, typically transmitted to the correct authorities 24 to 72 hours before arrival [9, 13]. The means of communicating this ETA varies and the accuracy of the ETA is not guaranteed to be within any margin of error. Ports, being very complex infrastructures with many moving parts, have to operate with certain uncertainties of vessels' arrival times and required demands and it is essential to plan port operations for a 24-hour period at a minimum to ensure smooth operations [14]. All uncertainties that can delay or alter any operations at one port are, therefore, a risk that can delay and disrupt all operations for not only the port in question but also all other ports that have any relations to the port.

There are many different practices for relaying the ETA to the correct authorities, for example via the AIS, phones and email. However, a study by Harati Mokhtari, Wall, Brooks, and Wang [15] discovered that almost a majority of the transmitted AIS messages had faulty or inaccurate data for the destination and ETA. Another risk discovered with the destination and ETA, when the data was available, was that it was not updated and, thus, could lead to more problems if any party thought the information to be true. An incorrect or inaccurate ETA could mean that port authorities fail to plan their operations to handle incoming and outgoing traffic

within the expected time.

2.2 The Baltic Marine Environment Protection Commission

The Baltic Marine Environment Protection Commission, or Helsinki Commission (*HELCOM*) as it is also known, was formed in 1974 parallel with the *Convention on the Protection of the Marine Environment of the Baltic Sea*. The participating parties are Denmark, Estonia, the European Union, Finland, Germany, Latvia, Lithuania, Poland, Russia and Sweden with the goal to protect the marine environment from increasing exploitation and ensuring protection of the ecosystem in the Baltic Sea [16].

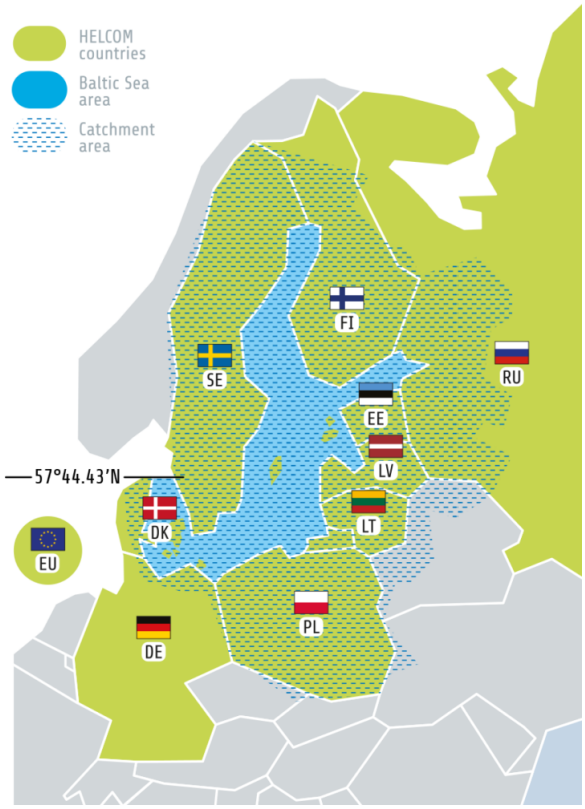

Figure 2: Area of the Baltic Sea of the Helsinki Convention, source `https://helcom.fi/about-us/`

In the early 2000s, as part of the HELCOM Copenhagen Declaration, it was decided that the Baltic Sea would be covered by an regional AIS network, to share AIS data in real-time with all participating countries. The Baltic Sea became the first region that is covered by AIS stations, allowing for real-time tracking and monitoring of all vessels equipped with an AIS in the region.

2.2.1 HELCOM dataset

The HELCOM dataset consisting of AIS data, covers the period from January 2009 to December 2019. Each month for each year is stored as its own comma separated value *.csv* file.

The AIS data in the HELCOM dataset have already been processed from what is originally transmitted from the AIS. All rows in the dataset has been made uniform to include the same data; every entry in the dataset contains exactly the same features. This has been possible by merging many different databases into one and this has also made it possible to cover the whole operating area of HELCOM, which includes all of the Baltic Sea, as seen in Figure 2.

2.2.2 Description of HELCOM dataset features

The HELCOM dataset has twelve unique features for each entry, as described in Table 3. The features are static, dynamic and voyage-related AIS data merged from multiple sources.

The combination of these features gives information about the vessel's identification, position, time stamp, trajectory, draught and physical size.

Name	Description	Value
timestamp	Unix epoch time in milliseconds when AIS message was created	Min 1230768000000
mmsi	Maritime Mobile Service Identities, unique for each vessel, can change	9 digit identifier
lat	Latitude position when AIS message was generated	Coordinate in WGS 84
long	Longitude position when AIS message was generated	Coordinate in WGS 84
sog	Speed over ground in knots	0.1 knot resolution
cog	Course over ground in degrees relative to true north	0.1 degrees
draught	Vertical distance from waterline to the keel	0.1 meters
dimBow	Reference point for position of positioning system on the vessel	Meters from bow
dimPort	Reference point for position of positioning system on the vessel	Meters from port side
dimStarboard	Reference point for position of positioning system on the vessel	Meters from starboard side
dimStern	Reference point for position of positioning system on the vessel	Meters from stern
imo	Unique identifier for each vessel, does not change	7 digit identifier

Table 3: Variables in HELCOM data set and description.

2.2.3 Statistical information

The HELCOM dataset covers a large window of time and gives an insight into historical vessel movements on the Baltic Sea. The dataset has been created by merging many databases and processing the data to create a uniform dataset for the whole period. In this dataset, it is possible to find vessels' movements on the Baltic Sea, where they are going and where they have been, by grouping the data by unique vessels and then order the data for each vessel in chronological order. It is not given that a vessel will have a contiguous timeline without gaps, this due to the possibility when a vessel has travelled outside the area covered by the AIS or has turned off its AIS for any reason. These gaps can be many months, even years, long and do not impact the quality of the routes. Smaller gaps in the timeline of a vessel do, however, impact the quality of the routes. There are many reasons why vessels' timelines can have smaller gaps, less than a couple of hours, and all of them will negatively impact the possibility of finding good routes. Some of the most likely reasons are malfunctioning AIS, the AIS has been turned off or the data received have been faulty and not stored.

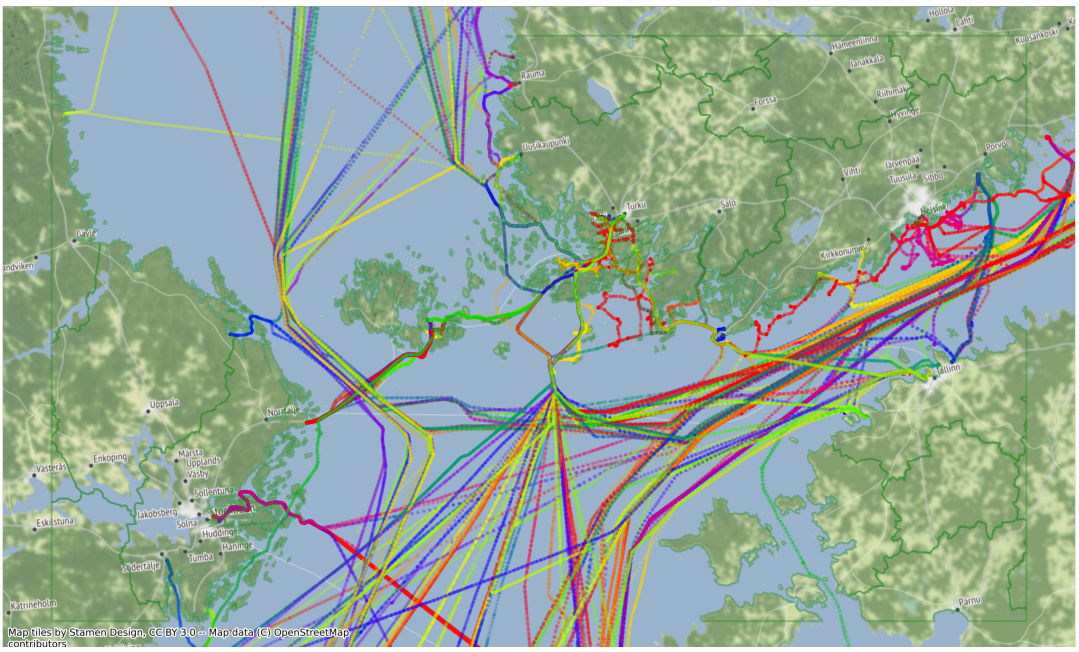

Figure 3: Small window of time of raw AIS data from the HELCOM dataset. Coloured by unique vessels.

Figure 3 shows a small window of time from the HELCOM dataset of all AIS data. Only a fraction of these data is viable for further processing, but a first pass over the whole dataset is required to capture all possible vessels for each month.

The average speed over ground for routes going to the Port of Naantali, seen in Figure 4, indicates an even distribution of vessels travelling about 14 knots. This is expected from the types of vessels that are equipped with an AIS and have been recorded in the dataset.

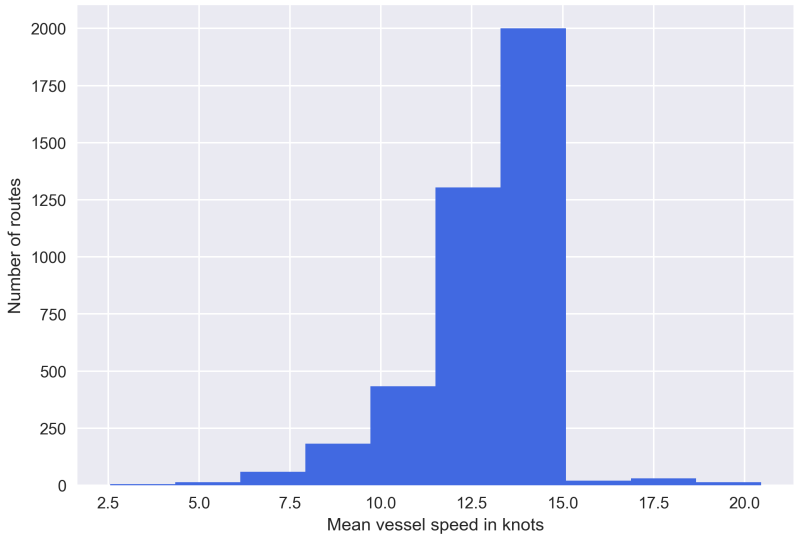

Figure 4: Average vessel speed for a route for all routes used, from the period of 2009 to 2018.

The time difference between AIS messages does not correlate to what raw AIS

transmission rates should be, see Table 2. A concentration of time difference between one and four minutes is another indication of pre-processing done by HELCOM, as the timesteps compared are from windows of time where vessels have been under way during a route.

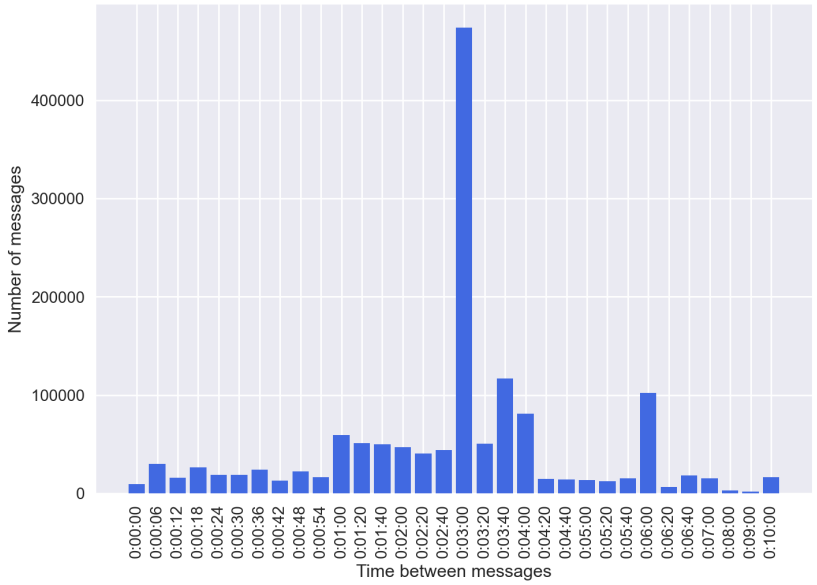

Figure 5: Time difference between AIS messages for any routes found during a five year period. All time differences greater than ten minutes have been collected in the same bar.

Vessels' physical features and the grouping of different vessels by size. In Figure 7, an example of one year of unique vessels which have visited the port of Naantali can be seen. Classifying vessels according to the physical size of the vessel has shown to be an efficient way of discerning different vessels' ability to manoeuvre [6]. It is expected that vessels of similar dimensions manoeuvre in a similar fashion. The classification was done by estimating different groups of vessels from their length and width, after which they were classified into classes according to the boxes seen in Figure 7.

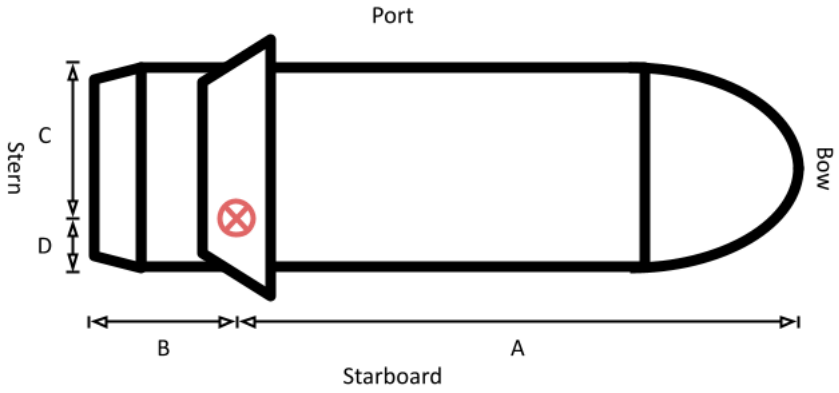

Figure 6: Vessel class definition from HELCOM data, the positioning system marked with red. A is *dimStern*, B is *dimBow*, C is *dimPort* and D is *dimStarboard*.

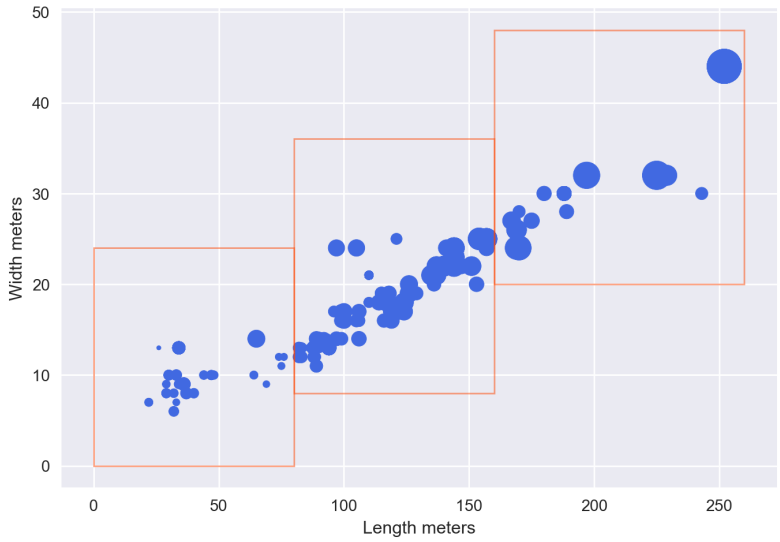

Figure 7: Defined vessel classes and distribution of all unique vessels for the year 2015. Size of the circle is related to the draught of the vessel.

2.3 Port of Naantali

The Port of Naantali is a rather important and busy port in the Baltic Sea and it is considered one of Finland's busiest ports [2]. With a reported total of over 8 million tonnes of cargo passing through the port and over 1,000 port calls during 2020, its logistical importance is of significance [17]. For the years 2018 through 2020, the port handled on average 5.8% of all of Finland's transports by sea and, combined with the port at Kilpilahti, handled all of Finland's crude oil transport [18].

This is also recognised in the HELCOM data by the amount of the data that has any connections to the Port of Naantali.

Year	**2009**	**2010**	**2011**	**2012**	**2013**	**2014**	**2015**	**2016**	**2017**	**2018**
% of data	8.21	7.16	7.30	13.19	12.30	12.30	12.30	10.17	10.87	10.61

Table 4: Total amount of vessel data with connection to the Port of Naantali for each year of the HELCOM data.

The percentage of the data in Table 4 is the complete timeline for each vessel that has at any point during the years visited the Port of Naantali, including other voyages that are not going and coming from the Port of Naantali. Only a fraction of these data is actual viable data for the routes, as described in 4.2.1. Still, the fact that 10% on average of all data has any connection to the Port of Naantali further indicates its importance for the Finnish shipping industry and corroborates it being chosen of all possible ports.

The port is defined, for the scope of this work, by manually defining an area that covers the whole operational area of the port, to include all berths and the possible routes to approach the port. For all vessels except small pleasure boats, the Port of Naantali is approachable from one direction, see Figure 8. The bounding box in red in Figure 8 is what defines the area of when a vessel has entered the port *i.e.* what defines a port. For the scope of this thesis, it is the port authorities' responsibility to ensure that vessels are allowed to berth as close to the ETA as possible or, if not possible, communicate another ETA for the vessel so it can adjust its speed and manage just in time arrival (JIT).

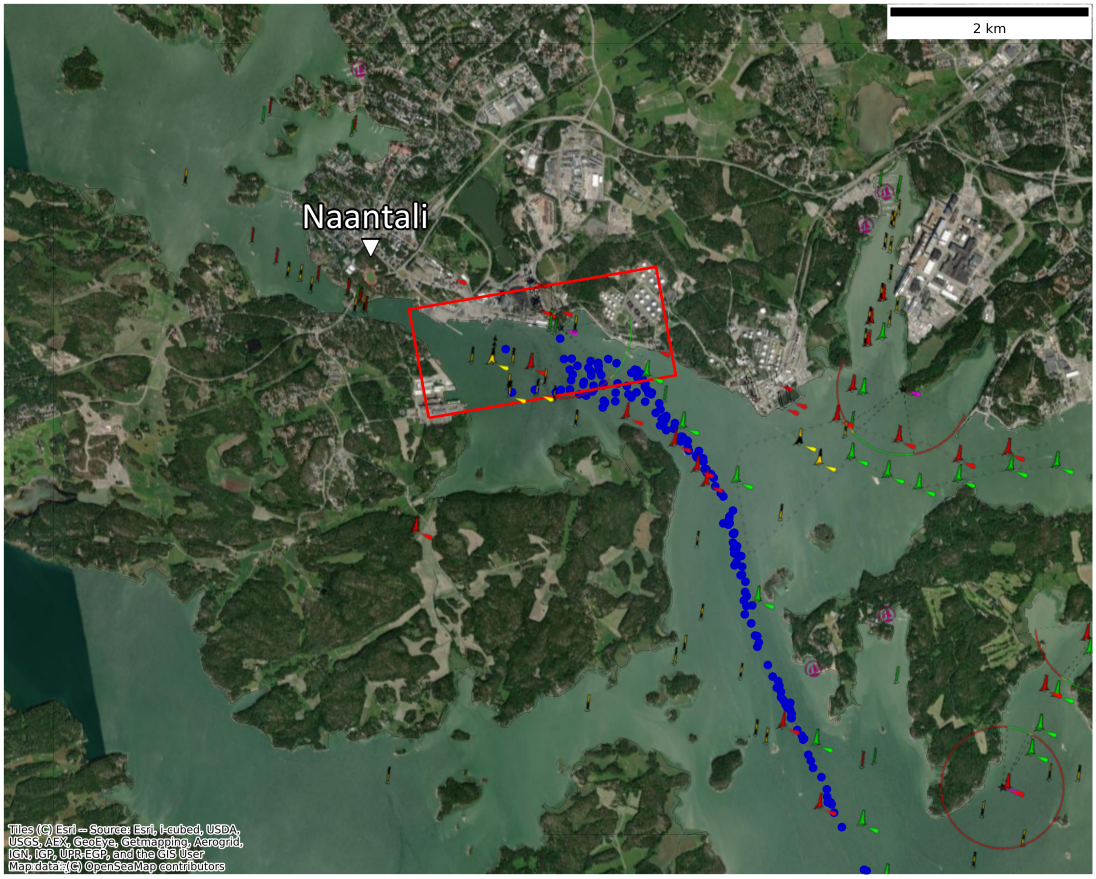

Figure 8: Defined bounding area for the Port of Naantali outlined in red. A small sample of the incoming routes shown in blue.

3 Artificial neural networks

This section will the fundamental theory to understand the basics of neural networks and a deeper understanding of recurrent neural networks.

Artificial neural networks was first conceptualized in the 1940s by American psychologist Warren McCulloch and mathematician Walter Pitts, which proposed a model known as the M-P model. The M-P model was a simple functional logic device with limited functionality, but still considered an important start of the theoretical research of neural networks. Frank Rosenblatt, another American psychologist, proposed in 1957 a perceptron model based on the M-P model which created the first true neural network that can learn by adjusting the properties of the network. After a book called *Perceptrons*, published in 1969 by Marvin Minsky and Seymour Papert, the field of artificial neural networks was coerced into a hiatus due to the authors implicating that the perceptron model is limited to simple linearly separable tasks and, therefore, the whole network could not understand the logical relationship of XOR and other linearly inseparable tasks. In the 1980s, American physicist John Hopfield proposed a new type of neural network that sparked new interest in studying neural networks, called a discrete Hopfield network. The potential of the Hopfield neural network encouraged scientist to further develop neural network, which led to the neural networks that exists today [19].

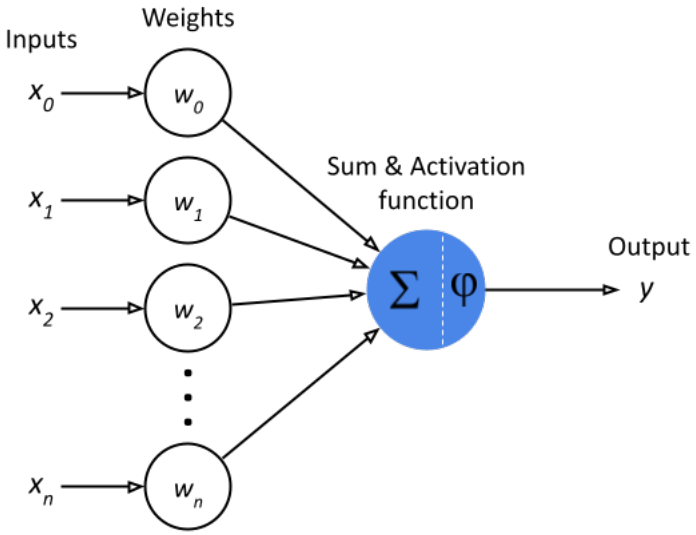

Figure 9: An artificial neuron.

Artificial neural networks (ANN) is an artificial representation of the biological brain. The biological brain consists of neurons which communicate with electrical impulses that activates the neuron, creating a response that is the collected sum of all incoming signals. A neuron in the biological brain has many inputs and one output, and the output depends on what the neuron has learned to interpret all

the incoming signals as. Combining many neurons creates a network that have the ability to learn by adapting the responses of all neurons in the network. It is the process of learning what to interpret a collection of incoming signals as and output a correct response that makes the network perform well when given a task which the network has learned to perform. A untrained network will have no knowledge of how the neurons should interpret the incomings signals and will therefore be as good as a random guess at the result. The goal of the training is for the network to learn the underlying structure of the data and by learning that, be able to make predictions that are accurate given data the network is not familiar with [20]. It achieves this by adjusting the weights of each input for each neuron and adapting the threshold function, typically done with a technique called gradient descent.

A simple artificial neuron, as seen in Figure 9, consists of n number of inputs x. These inputs are multiplied with each associated weight w after which they are summed together and further processed by the activation function. The operation can be expressed as

$$u = \sum_{n=1}^{n} w_n x_n$$

with u being the value that is fed to the activation function. The activation function is what determines at what output u the neuron activates. For example, if the activation function is a step function, anything above a certain threshold φ creates a response otherwise the neuron is inactive, and the total input is 0.6 with a threshold of 0.5 the output y of that neuron would be 1. This type of unit is called a threshold logic unit (TLU). There are many threshold functions, each with its strengths and weaknesses, and choosing the correct one is dependant on the task the neural network should learn to solve. Step functions can work for classification tasks but does not work for regression tasks where a continuous range of values is the desired output.

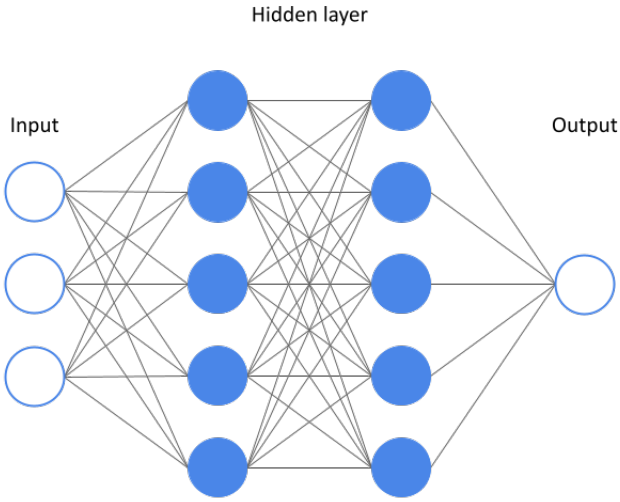

Figure 10: Simple artificial network of three input neurons, two hidden layers with five neurons each and one output neuron.

The TLU introduced in this section is one of the simplest form of a artificial neuron and is only suitable for a small set of tasks. A more advanced type of unit is required for the work of this thesis and this type of unit is introduced in section 3.1.1.

3.1 Recurrent neural networks

Recurrent neural networks (*RNN*) are a class of neural networks that are well suited for time or sequence dependant data. RNNs achieves this by implementing a internal state, *memory*, that can remember temporal dependencies in the data.

Regular feedforward neural networks can predict independent data points that have no relation with the order of the data, for example, classifying images or recognizing patterns. When the order of the data has an impact on the final result, for example, weather forecasts are dependant on what the weather has been the past days or knowing the past trends when predicting stock prices for a specific stock, feedforward neural networks have no way of storing information about past events and can only make a prediction on the current data with no *priori* knowledge [21, 22, 23]. Feedforward neural networks process the input to produce an output by a finite series of neurons with no feedback coming from the output to the input [20]. If the network could store information that the weather has been warm the past days, it is more likely to predict the next day is also going to be warm. Regular feedforward neural networks also have the limitation of not being able to process variable length input data.

RNNs are an extension of regular feedforward neural networks that implements hidden feedback loops within the hidden layers. The feedback loop can be unrolled,

which gives a representation similar to a regular feedforward neural network, but each layer is instead a cell that represents one timestep. The depth of a single RNN neuron is equal to the number of timesteps used [24]. Figure 11, visualizes one recurrent neuron with three timesteps unrolled. RNNs can, contrary to feedforward neural networks, map multiple inputs to multiple outputs, one input to multiple inputs and multiple inputs to one output. For example, x_1, x_2 and x_3 can output only y_3, which can be utilized when the result is dependant on a sequence of inputs.

The feedforward operation through the network is used to get the output of the network, the result passing data throught the network. While training, the output of the network is compared with the actual correct value, giving the error of the network. In regular feedforward neural networks, backpropagation is the process of moving backwards through all the layers of the network. For every layer, the partial derivative is calculated of the error for each respective weight, thus, giving the values required for optimizing the weights using a process called gradient descent. Gradient descent is a function that minimizes a given function and the network wants to minimize the error by increasing or decreasing each respective weight, which is achieved by using gradient descent. Because each RNN neuron implements a feedback loop, regular backpropagation is not viable for updating the weights in a RNN, instead a similar method called backpropagation through time ($BPTT$) is utilized. BPTT is an extension of backpropagation, that is able to calculate the derivatives for the RNN neuron. The process can be thought of as unrolling the RNN neuron, which yields a representation similar to that of a layered feedforward network, but instead of layers there are cells. This unrolled representation can then be used to calculate the derivatives which are used to optimize the network by gradient descent.

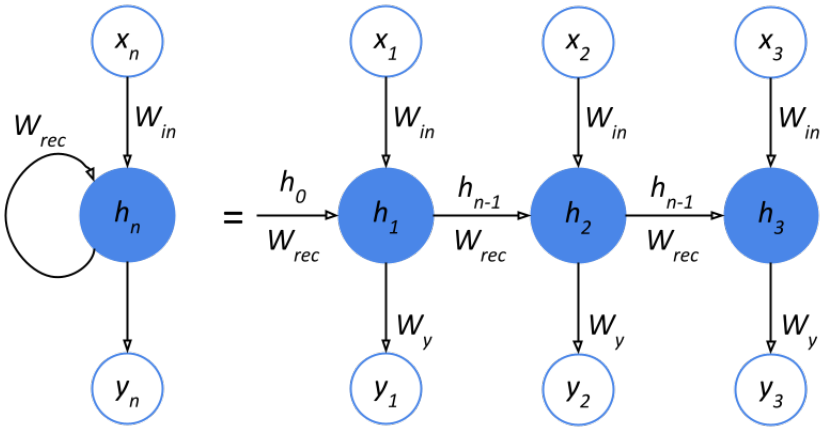

Figure 11: Single RNN neuron with feedback loop on the left and an unrolled RNN neuron with three timesteps on the right.

All deep neural networks suffer from gradients that can either explode or vanish

[24]. A deep network is any network with multiple layers between the input and the output. The problem is called vanishing or exploding gradients, and is noticeable in regular RNNs, each neuron is a stack of cells similar to a feedforward neural network with multiple layers, making each neuron similar to a deep neural network. Using gradient descent and BPTT to update the weights for each layer and every temporal chain of cells throughout the network, can cause the gradients to either converge to zero or infinite, which consequently, halts the learning process.

The following notation is used in Figure 11 and for the following equations, x_n denotes the input at timestep n, y_n denotes the output at timestep n, h_n stores the state of the cell, which, gets passed to the next cell, and h_0 is set to zero, W_{in} denotes the weight for the input and W_{rec} denotes the weight for the hidden units, W_y denotes the weight for the output. The bias is left out of the equations to simplify them, but would be added to the inner expression in equation 1 as b_n.

The output of the cell h at timestep n is calculated as below, φ denotes any activation function:

$$h_n = \varphi\left(W_{rec}h_{n-1} + W_x x_n\right) \tag{1}$$

The output y at timestep n is calculated as:

$$y_n = \varphi\left(W_y h_n\right)$$

Calculating the output of the third cell can, therefore, be expressed as:

$$y_3 = \varphi\left(W_y\varphi\left(W_{rec}\varphi\left(W_{rec}\varphi\left(W_{rec}h_{n-3} + W_x x_{n-2}\right) + W_x x_{n-1}\right) + W_x x_n\right)\right)$$

Product rule to find the gradient for weight W_{rec}:

$$\frac{\partial\left(W_{rec}h_{n-1}\right)}{\partial W_{rec}}$$

The output of y_n can be generalized as a series of functions, one for each layer in the network:

$$y_n = f\left(g\left(h\left((x)\right)\right)\right)$$

Using the chain rule to calculate the gradient at the first layer W_1 can then be expressed as:

$$\frac{\partial y}{\partial W_1} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial h}\cdots$$

The gradient of the error for timestep k, over T timesteps, can then be expressed as.

$$\frac{\partial E_k}{\partial W} = \frac{\partial E_k}{\partial h_k}\frac{\partial h_k}{\partial C_k}\left(\prod_{n=2}^{k}\frac{\partial h_n}{\partial h_{n-1}}\right)\frac{\partial C_1}{\partial W} \tag{2}$$

An alternative way of expressing the gradient error on the k-th timestep, for n timesteps, using the derivative of h_n.

$$\frac{\partial E_k}{\partial W} = \frac{\partial E_k}{\partial h_k}\frac{\partial h_k}{\partial c_k}\left(\prod_{n=2}^{k}\varphi'\left(W_{rec}h_{n-1} + W_x x_n\right)W_{rec}\right)\frac{\partial h_1}{\partial W}$$

To understand the problem of vanishing and exploding gradients, the equation above can be thought of as a series of repeating functions where the same values are repeatedly multiplied. For the vanishing gradient problem, the weight W_{rec} is multiplied multiple times and if the weight is very small, the value will only decrease more every time it is multiplied. The same is true for the exploding gradient problem, only the value getting multiplied will quickly approach infinity.

3.1.1 Long Short Term Memory

To solve the problem with long term dependencies and mitigating the vanishing or exploding gradients, Hochreiter and Schmidhuber introduced in 1997, an alternative to regular RNN units called Long Short Term Memory unit or simply LSTM [25].

LSTMs are a unit that replaces the regular cells in an RNN. Compared to a regular cell, LSTM are quite complicated, with multiple pointwise operations, dark blue circles, and learned network layers, light blue rectangles, see Figure 12. The main component of the cell, which enables the cell to perform better than regular RNN cells, is the horizontal line running through the cell at the top, the *cell state*. It allows for information to flow through the cell mostly unchanged, only minor linear operations interacting with the cell state as it moves through the cell. The cell can alter the state of the cell state by a set of carefully regulated gates. The gates are made up of one sigmoid neural net layer and one pointwise multiplication operation. In total, there are three of these gates in the cell and these gates are called the *forget gate*, *input gate* and *output gate*.

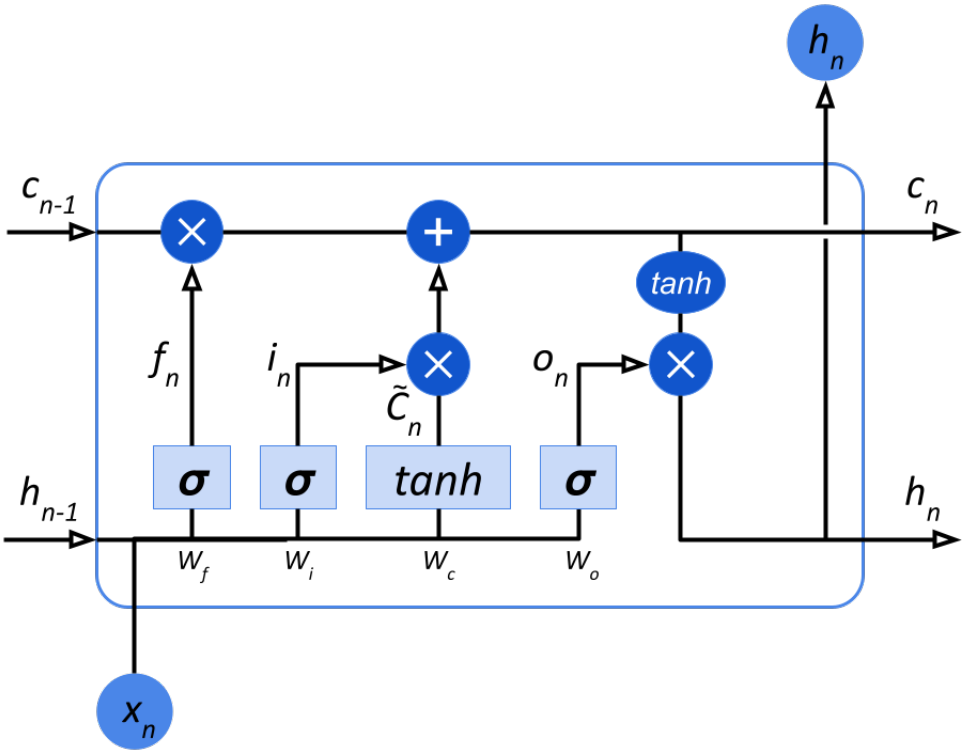

Figure 12: One LSTM cell with the recurrent cells omitted, source [26].

The forget gate is the first operation in the cell and decides what information to discard given new information. The notation $[h_{n-1}, x_n]$ is used for the input vector.

$$f_n = \sigma\left(W_f\left[h_{n-1}, x_n\right] + b_f\right)$$

After the forget gate has decided on what information to discard, the cell decides what new information to store in the cell state. This step is divided into two steps, first, a sigmoid layer decides which values will get updated, handled by the input gate, in Figure 12 this is denoted by i_n. Then, a new vector of possible values is created by the tanh layer, denoted by $\tilde{C}_n$.

$$i_n = \sigma\left(W_i\left[h_{n-1}, x_n\right] + b_i\right)$$

$$\tilde{C}_n = \tanh\left(W_c\left[h_{n-1}, x_n\right] + b_C\right)$$

The values i_n and $\tilde{C}_n$ are then combined to update the state C_{n-1} to C_n. This step is where the cell discards the old information to replace it with the new, according to what was deemed best by the previous steps.

$$C_n = f_n C_{n-1} + i_n \tilde{C}_n$$

Finally, the last step is to output the new information.

$$o_n = \sigma\left(W_o\left[h_{n-1}, x_n\right] + b_o\right)$$

$$h_n = o_n \tanh\left(C_n\right)$$

The LSTM cell solves the vanishing and exploding gradient problem by making use of the gated internal structure of the cell. The symbol $\cdot$, in equation 3, denotes pointwise multiplication and the symbol + denotes pointwise addition.

$$C_t = C_{n-1} \cdot f_n + \tilde{C}_n \cdot i_n \tag{3}$$

The derivative of equation 3, is equal to the following expression. Note, the + symbol is not the pointwise operation for the following equations.

$$\frac{\partial C_n}{\partial C_{n-1}} = \frac{\partial f_n}{\partial C_{n-1}} C_{n-1} + \frac{\partial C_{n-1}}{\partial C_{n-1}} f_n + \frac{\partial i_n}{\partial C_{n-1}} \tilde{C}_n + \frac{\partial \tilde{C}_n}{\partial C_{n-1}} i_n \tag{4}$$

The gradient for the LSTM cell can be expressed as an additive gradient as below.

$$\frac{\partial C_n}{\partial C_{n-1}} = I_n + J_n + K_n + L_n \tag{5}$$

Where the equations 6, 7, 8 and 9 corresponds to each element in equation 5, which are the derivatives for the four terms in equation 4.

$$I_n = \sigma'\left(W_f\left[h_{n-1}, x_n\right]\right) W_f o_{n-1} \cdot \tanh'\left(c_{n-1}\right) c_{n-1} \tag{6}$$

$$J_n = f_n \tag{7}$$

$$K_n = \sigma'\left(W_i\left[h_{n-1}, x_n\right]\right) W_i o_{n-1} \cdot \tanh'\left(C_{n-1}\right) \tilde{C}_n \tag{8}$$

$$L_n = \sigma'\left(W_c\left[h_{n-1}, x_n\right]\right) W_c o_{n-1} \cdot \tanh'\left(C_{n-1}\right) i_n \tag{9}$$

The LSTM states gradient at timestep k can then be expressed as below, using the equation for calculating the gradient of the error, see equation 2.

$$\frac{\partial E_k}{\partial W} = \frac{\partial E_k}{\partial h_k} \frac{\partial h_k}{\partial c_k} \left(\prod_{n=2}^{k} \left[I_n + J_n + K_n + L_n \right] \right) \frac{\partial c_1}{\partial W}$$

Now the gradient is comprised of the forget gate's values, which allows the network to control the gradients values by adjusting the forget gate's parameters at

each timestep. By adjusting the parameters of the forget gate, at timestep $n+1$, so that the error gradient does not vanish, the LSTM unit is able to solve the vanishing gradient problem.

One problem that is not solved by regular RNNs or networks using the LSTM unit, is the distribution of the sampling rate of the data points. When the time between each sample is even, the networks thinks that the difference in time is equal, which is true for equally sampled data points. If the samples are not equally sampled, the time difference is thus not equal and, as time is linear, the impact of time on the sample is not linear and the model will have a hard time to understand the impact of each sample.

3.1.2 Phased LSTM

The Phased LSTM (*PLSTM*) implementation tested was developed by Francesco Ferroni and is available at `https://github.com/fferroni/PhasedLSTM-Keras` (11/03/2022)

4 Implementation

4.1 Libraries used for the implementation

TensorFlow is an open-source library created by Google for deep learning applications [27]. TensorFlow GPU enables TensorFlow to take advantage of the processing power of a GPU, which, inherently speeds up the training process.

Keras is a high-level application programming interface (*API*) of TensorFlow written in Python, other deep learning frameworks are available, however, TensorFlow has adopted Keras as the official high-level API making Keras the preferred API. Keras was developed to enable fast development from idea to execution. It was designed to be user friendly to give users a simple method of creating models.

NumPy is a library designed with multidimensional arrays in mind and is the preferred library for scientific programming [28]. NumPy implements many fundamental mathematical operations as a vectorized alternative, especially when operating on arrays. The reshaping of arrays is another key feature that NumPy implements very well and is vital when dealing with timeseries for RNNs.

pandas is an open-source library designed for data manipulation and analysis, and provides a plethora of built-in functions to process data [29]. As the data used for this thesis is stored in csv-files, the choice of *pandas* was clear from the start and further solidified due to the geospatial nature of the data and further on the choice of using GeoPandas.

Shapely is an open-source library for Python to manipulate geometric objects. The basic objects are points, lines and polygons.

Geopandas is an open-source library that extends *pandas* datatypes and can operate on Shapely objects, and provides vital functions when working with geospatial data [30]. It is written in Python and allows for almost seamless development between *pandas* and GeoPandas. The library can be quite a challenge to install, especially when other libraries that requires the same libraries are installed or needs to be installed. GeoPandas utilizes *pandas* datatypes to store data and Shapely to operate on the data, for example, coordinates are simply points in a space and areas can be represented as polygons by simply connecting the points with lines. These fundamentals allows the representation and manipulation of geospatial data.

4.2 Data pre processing

The years 2009 through 2018 are chosen to train and evaluate the performance of the model. The year 2019 was omitted due to unusual file sizes, which led to the belief that the files for that year are not complete. All these files total approximately 130 gigabytes, which will introduce some limitation on how much of the data is

processed at once. The process of finding possible routes in this raw data for further processing and validation involves finding all vessels that have visited the port of interest at any point. Due to the file sizes and amount of data for each file, this process has to be split into smaller chunks of time windows, *i.e.* the number of consecutive months processed at once. With the possibility of vessels starting a route at the end of a month and arriving at the beginning of the next month, larger time windows are preferred, but memory limitations impeded the possibility of very long time windows. Splitting the year into quarters proved to be a good compromise computationally.

The first step involves finding which vessels have visited the port of interest, the port of Naantali, for each month. This step is computationally quite time-consuming, since every row in each file has to be checked, except rows of a previously known vessel. Only once a vessel has been identified to have visited the port of interest it can be skipped, else the row has to be checked. It is impossible to known at what point in time a vessel has visited the port of interest, so every row has to be checked and since a vessel could have made only one visit to the port for the whole period covered, it is unnecessary to store the whole timeline of said vessel for further processing. Rather, only the timeline of when the vessel has visited the port can be fetched, which drastically decreases the required amount of data. The process involves checking each row's latitude and longitude and whether these coordinates are within the bounding box, as seen in Figure 8. This process has to be done once for each file, after which the unique identifiers, *imo*, for every vessel can be stored for future use. Every month can have more than 30 unique vessels, a quarter can have around 80 unique vessels, and the whole timeline is extracted for all vessels for the quarter.

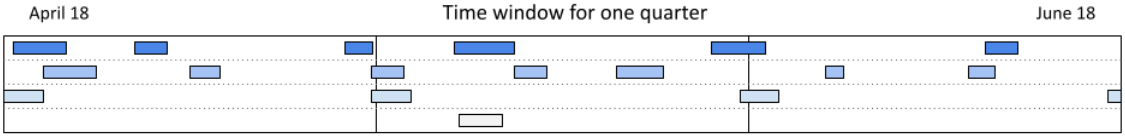

Figure 13: Example of one quarter of the year 2018, a *time window*, and from this time window each unique vessel's complete timeline is extracted for further processing. The coloured boxes represent raw AIS data for four unique vessels. Does not represent actual data, rather visualizing the process of finding relevant data in the raw HELCOM dataset.

From the complete time window seen in Figure 13, only a fraction of the data is actual viable routes going to the port of interest. Merging the complete time window with AIS data for every vessel generates a timeline for each of the unique vessels discovered in the first step. Within that timeline exists routes going to the

port of interest and the next section describes how these routes are extracted and validated.

4.2.1 Algorithm for extracting routes from dataset

A route is defined as a series of consecutive AIS messages where the first message in the series is the starting point for a vessel travelling to a port of interest. The last message in the series is the first messages when the vessel has entered the port of interest bounding area, see Figure 8. The routes do not need a specific starting point, as this point bound to any position; rather it will be chosen from one of two possibilities. The end of the route will be the same for all vessels, as the goal is to find all routes coming from somewhere travelling to the port of interest. A good route will also not have any gaps of time during the route.

To find routes coming from anywhere and going to an area of interest or port an iterative approach is used. With a complete historical timeline of one vessel, the routes can be discovered by traversing the timeline in reversed chronological order. The end of a route, *i.e.* the start of the search, is the first point where the vessel is leaving the bounding box. From this position, traversing the timeline, in reversed chronological order, until a predefined condition is met, will give the section of a vessel's timeline which equals a route coming from somewhere travelling to an area of interest.

The condition for when the start of the route, end of the search, is met, is whenever the duration of the route has exceeded 48 hours or the vessel's speed over ground is zero, standing still, for more than 5 consecutive transmitted messages. The 48-hour time limit was defined for the reason that the maximum travel duration in the Baltic Sea is below 48 hours for a vessel moving at the mean speed discovered and vessels travelling from outside the Baltic Sea have reached Kattegat and the edge of the Baltic Sea Area seen in Figure 2 at this point in time. The reason behind checking that multiple transmitted messages have recorded the speed to be zero is to guarantee that the vessel has actually stopped and not slowed down for other reasons. This condition can still introduce routes that have not started from another port but will minimize these routes.

Algorithm 1: Find all routes going to an area of interest

Input: DataFrame for one vessel sorted in descending time, a *timeline*

Result: List R of all routes found

The algorithm searches in reverse order of time from reaching the destination until the start of the route;

```
foreach row in DataFrame do
    index ← Keep track of current row;
    if route found then                    // Only true after finding first route
        Start searching from first unknown point;
        Skip rows until index = start;
    if current point is in PORT then
        foreach row in DataFrame starting from index do
            if point is outside PORT then      // Vessel is entering the port
                end ← First point reaching the destination;
                foreach row in DataFrame starting from end do
                    if start of route reached then
                        start ← Current row;
                        R ← Save route from end to start;
    /* When a route has been found and saved start search
       again from the last not visited point.                */
```

Route validation rules were defined to give an efficient way of discarding any routes that would impact the model's performance and routes that were off no interest for the scope of the thesis.

All routes that did not travel for more than eight hours from start to finish were discarded. The reason for this rule comes from the fact that all vessels will be required to travel through the same area for the final part of the voyage, and the model should perform well in this area, even though these routes are discarded.

A minimum distance travelled rule was used to discard any routes that travelled less than 200 kilometres, that is the total route distance and not how far away from the port the vessel travelled. To navigate out from the port of Naantali and the largest bodies of islands outside of Naantali takes approximately 70 kilometres following the fairway. Moving at 14 knots the time to cover the distance takes just under three hours, however, the speed in this area will most likely be lower for most vessels due to the limited area to manoeuvre.

Any routes that have a time difference between two consecutive AIS messages greater than 12 minutes are also discarded. The reason for this is explained in more detail in Section 4.2.4.

Vessels that have returned to the port of Naantali or are returning to the area around Naantali are also discarded. This combined with the minimum distance travelled required will ensure that, for example, pilot vessels and other leisure vessels will not be in the final training data. Vessels travelling back to the area are not of interest and could impact the model negatively, as these vessels will probably have an abnormal voyage compared to cargo vessels.

Routes that have faulty data, *i.e.* NaN values or other bad values, and which cannot be inferred from the data in any other way are discarded. Examples of this was found where the draught was not recorded only for a small section of the route but could be inferred from the parts where it was recorded. If the values could not be inferred at all, the route was discarded completely.

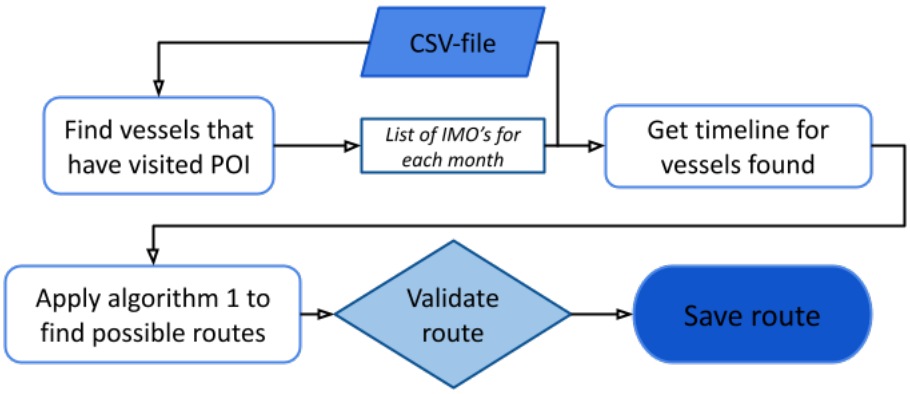

Figure 14: Getting routes.

After all of the steps and validations above the final number of routes which fulfil the requirements defined is 3,973. The number of possible candidate routes was 13,955, 70% of all routes did not meet the requirements and were therefore discarded. This further indicates the issues with the veracity of AIS data. The number of routes that could be possible valid candidates could be higher by generating data for missing gaps on the timeline of the routes, when the gap is below a certain threshold. For the scope of this work, to utilize historical AIS data provided by HELCOM and create a model that can predict the ETA for vessels in the Baltic Sea, the number of routes found does prove it is possible, but with limitations.

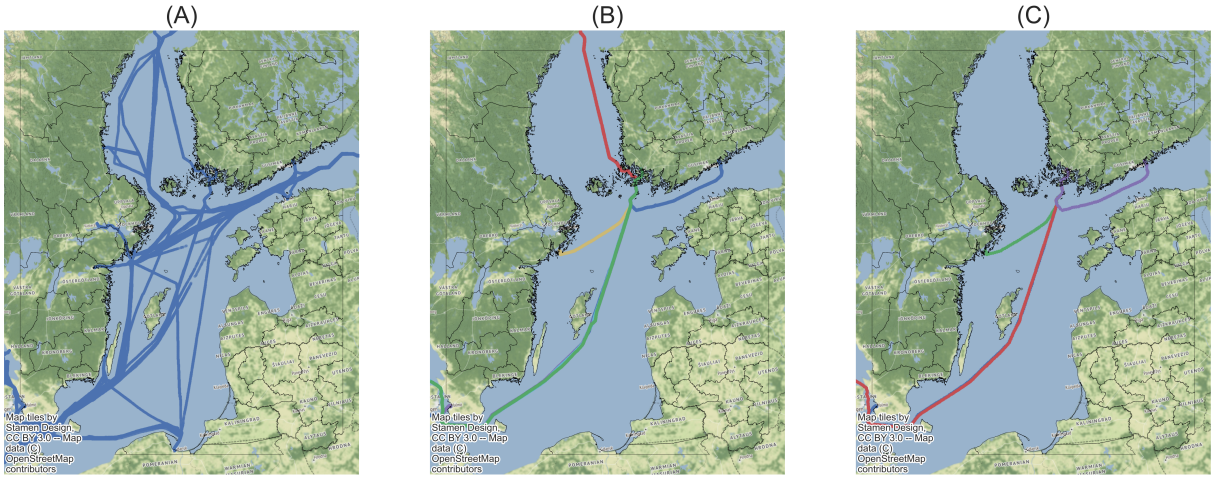

Figure 15: (A) Complete timeline for one vessel in 2013. (B) All possible routes for the same vessel. (C) All validated routes for one year from one vessel which are used for training.

Figure 15 illustrates these limitations where routes have to be discarded for not meeting the required validation rules. Also with the data in Table 4 in mind, the amount of data of the total timeline that make up the good routes is only a fraction of it.

4.2.2 Coordinate accuracy

The latitude and longitudes in the HELCOM dataset have an accuracy of six decimal places, 0.111 metres. This accuracy is not required as the difference in time between a lower accuracy will be within a possible prediction error.

The level of scaling the coordinate accuracy with, was chosen to a decimal degree of three places, that is 0.001 which is 111 meters. The accuracy of the vessel's location will be within approximately 100 meters.

Degrees	Distance
1.0	111 km
0.1	11.1 km
0.01	1.11 km
0.001	111 m
0.0001	11.1 m
0.00001	1.11 m
0.000001	0.111 m

Table 5: Coordinate accuracy by decimal places in decimal degrees for latitude and longitude, from [31]

AIS messages transmitted within approximately 100 metres of each other will be thought of as sent from the same location using this scaling level. With a mean speed

for the routes found and used, see Figure 4, of approximately 14 knots (25 km/h) and a time difference between messages of 10 minutes a vessel will have moved 2.2 nmi (4.1 km) during this time. With a positional accuracy of approximately 100 metres, this distance can vary but on the open sea a difference in ± 100 metres does not impact the overall time it will take to reach the destination.

4.2.3 Feature selection

The features chosen for training the model on were decided by what is available in the HELCOM dataset, what is possible to infer from available data and what has been proven to be efficient features by previous studies [8, 6].

Most of the features available in the HELCOM dataset were used for the model; latitude and longitude for the position of the vessel at a given point in time, sog to indicate the vessels moving rate, cog for where the vessel's trajectory is heading, draught to indicate the vessel's physical capabilities in combination with the vessel class feature, which is a combination of features, see Figure 6 and, finally, the distance to the destination calculated from the historical route the vessel has taken.

Using the distance left to the destination as a feature does mean that the problem to be solved is, in its simplicity, learning the time it will take to traverse the distance left at the current speed. However, there are many factors that will impact the time it will take to reach the destination which cannot be learned from only the distance to travel and the speed. It is these factors that could be learned by training on historical data and improve the prediction accuracy compared to other simpler solutions.

The target to predict is time to destination (TTD) which is in minutes. The TTD prediction is added to the current time and, thus, gives an ETA. The TTD is calculated as the difference between the current time and what time the vessel has arrived at port.

4.2.4 Time series data windowing and time distribution

With the routes validated, a final processing step is required to generate the time series data that is fed to the network. Since regular RNN suffers from poor performance when the data are unevenly sampled in terms of time difference between samples, it is vital to normalize the time difference between time steps. The nature of the AIS and the HELCOM dataset means that it is only viable to normalize the sampling interval to a certain degree.

A compromise with the time difference between messages was achieved by as evenly as possible sample data with longer time intervals, discarding messages in that interval, and by this achieving an improved time series with preferable time difference normalization. In this improved time series, the difference between messages is on

average at most three minutes. A lower limit of nine minutes was chosen to conform with what was discovered in Section 2.2.3 and Figure 5 and the upper bound of twelve minutes has been set when validating the routes. On average, two messages will be discarded, so the time difference is nine minutes. Shorter time intervals should be possible with raw AIS data, but for the dataset used the number of routes that were viable without generating data between messages were too few. Decreasing the longest allowable gap between AIS messages in a route discarded too many routes, *i.e.* a route with at any point a time difference between messages smaller than twelve minutes would have been rejected.

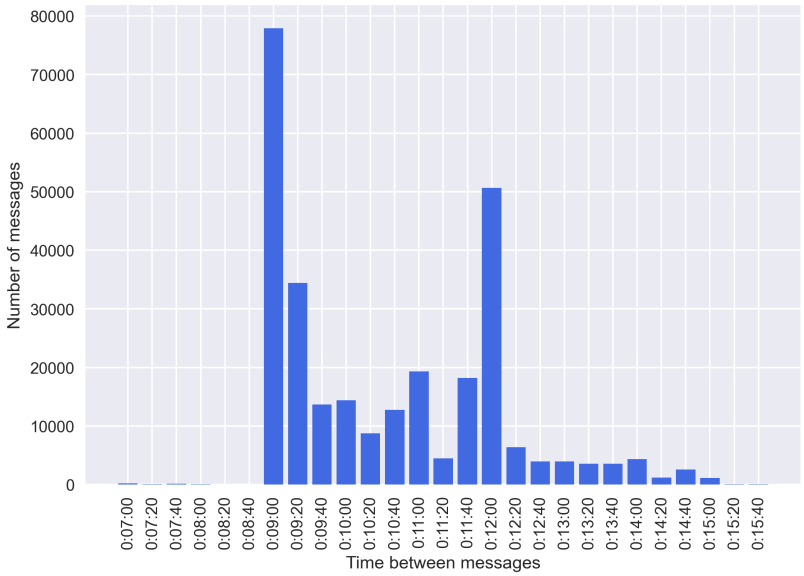

Figure 16: The distribution of time differences between messages for the normalized routes.

In Figure 16, the normalized time differences are in the range from nine to twelve minutes, with only some exceptions that have a time difference shorter than or longer than this range. The exceptions are for the most part due to the inconsistencies in the recorded time intervals, *e.g.* given one message at time zero t_0 with the following messages at t_{+4}, t_{+4} and t_{+7} (minutes from the previous message), the first two messages will not be more than nine minutes from the first and the third message will therefore be chosen as the next point in time, with a 15-minute difference.

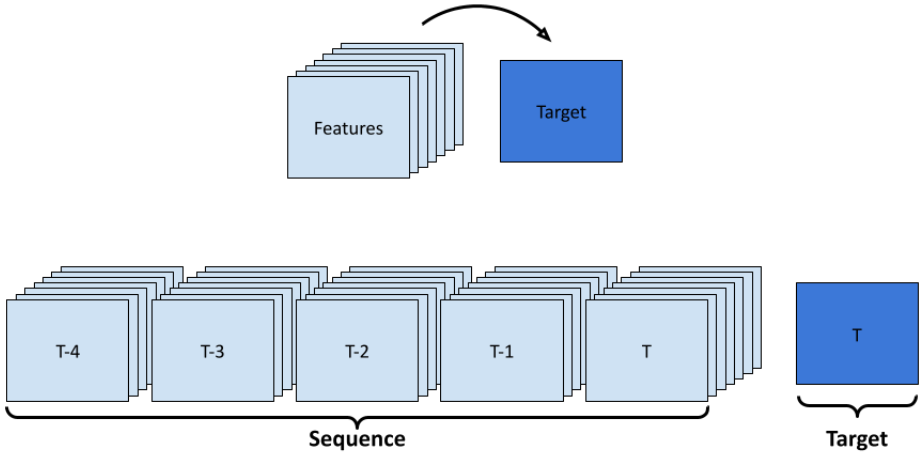

Figure 17: Time series sequenced data, with a time window width of five. Each timestep in the sequence has a time difference in the range defined.

With a route consisting of AIS messages with a time difference distribution in the range mentioned above, a time window of n timesteps is generated for the whole route. A time window is defined as n number of consecutive timesteps that create a window of time for the route. Each timestep in this time window consists of all chosen features and the target TTD. For all but the last timestep, the target is discarded and only the final timestep's target is set as the target to predict. Sliding the time window over the whole route, one timestep at a time generates a set of windows that each covers n timesteps.

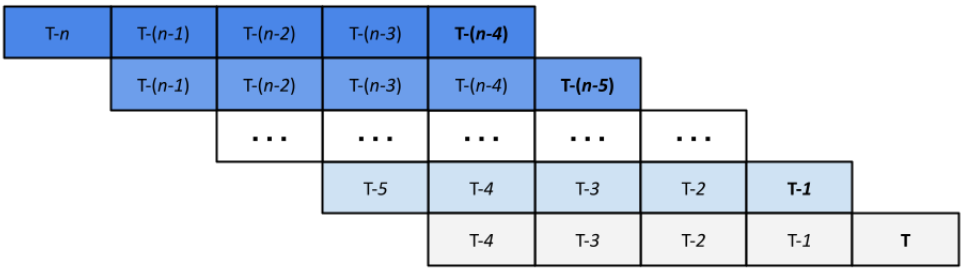

Figure 18: Sliding time window for a complete route starting from the end at $T - n$ and ending at the last timestep T. A prediction will be made for every last timestep in the time window, marked with bold.

A prediction will be made for the last timestep in the time window, which in terms of real-time prediction will require that for the first four timesteps, AIS messages received, a prediction cannot yet be made. After the fifth message a prediction can be made. This also means that any five historical AIS messages could be used, if they fulfil the requirements for the time window.

4.3 RNN model implementation

The neural network architecture implemented is a RNN with TensorFlow LSTM units. Earlier work has concluded that that the LSTM unit performed better than other alternatives, in a similar task of predicting the ETA [8]. The decision to not look into other alternatives was supported by these findings and the nature of how RNNs work, compared to regular feedforward neural networks or other types of neural networks.

Keras implements this architecture without much effort required from the user, the main work was put into manipulating and reshaping of the data, as the model requires a certain kind of data structure.

The final model used for training and testing consists of one input layer, two hidden layers and one output layer. During the development stages, single-layer networks and multi-layer networks were compared to each other, and the multi-layer networks performed better every time.

The input layer takes a vector of shape number of timesteps per time window t and the number of features per timestep f. This is the idea behind RNNs capabilities to utilize multiple timesteps in a sequence to predict an output, it takes multiple historical samples and learns the impact of previous samples on the final output.

The two hidden layers consists of 16 cells each and the first hidden layer has the argument `return_sequences` set to true, which is required when dealing with deep RNNs. The last hidden layer simply outputs the result of the layer to a single dense cell that outputs the prediction, TTD. The reason for 16 cells in each hidden layer is arbitrary, and while testing the impact of changing the model architecture, it was discovered that the performance difference was negligible between models with the same number of hidden layers but different number of cells.

The model was otherwise left unchanged from the default implementation, *i.e.* the activation was tanh, recurrent activation was sigmoid, recurrent dropout was 0, unroll was false and it uses a bias vector, these settings can be found on `https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM`. These settings were mostly chosen to utilize the cuDNN implementation, leveraging the GPU accelerated operations.

The optimizer used was the Adam algorithm with defaults settings. The reason for choosing Adam over any of the other available optimizers, was simply due to the performance and the optimizer is thought of, as one of the best performing with RNNs [8, 32].

4.3.1 Phased LSTM

Comparing the regular LSTM network with a Phased LSTM network, which sould perform better than LSTM when samples are irregularly sampled.

5 Results

The training process was done in an iterative process, by training on one complete route, from start to finish, thus training the model on complete routes that are divided into time windows as defined in Figure 17. This process was preferred to training on random sections of the route, as the model was confidently trained on the complete path of the route and no blind spots could be introduced.

The total number of routes discovered and validated was 3973. From this collection of routes, 35 were set aside for testing the performance, meaning there were 3938 routes used for training. The settings for the model is described in Section 4.3 and the training was set to 100 epochs.

The loss decreased the most during the first 50 epochs, after which the improvement stagnated, but no significant loss penalty was discovered when training for more epochs. However, a balance between decreasing the loss and the time it will take to train the model, without overfitting, is important to achieve a model that is feasible for real world applications. Therefore, the number of epochs was set to 100.

Metric	**Loss**
Mean	0.013
Minimum	0.000021
Maximum	0.059

Table 6: Performance of test routes.

The loss values in Table 6 does not provide any useful information without comparing the actual predictions to what the true value is. Inspecting the predicted time with the true value gives a better understanding of the loss values impact on the result and what loss value results in a good prediction. A loss value of 0.000x gives the performance wanted.

The prediction is processed one complete route at once for simplicity but the predictions are made one time window at a time and not the whole route at once. A prediction can be made for any section of the route with the time window corresponding to that section.

The maximum loss value for the model comes from a route that is abnormal compared to other routes discovered. A slow moving vessel travelling a route not visited by many other vessels.

The performance of the model leaves some room for improvement and it was thus discovered that the direction of travel has an impact on the route and stopped the model from reaching a performance sought after. Proving that this is the case with a *balanced* model and one *biased* model.

Mean loss for 35 training routes

Mean 0.013020759394151225

Min 2.0957821107003838e-05

Max 0.05850563943386078

Std 0.015516533321899

The idea to learn certain areas historical data and utilizing that to predict ETA is possible and might be an alternative to automatically update the ETA if not present or validating the accuracy of the broadcasted ETA with the prediction. The generic model is perhaps not there when measuring the overall performance, but training separate models on the overall direction from which the vessel is travelling from creates a model that can accurately predict the ETA within the constraints set.

5.1 ETA prediction

5.1.1 Biased training data by direction

East bound training with 277 train and 9 test

Average loss 0,002137 very close the the loss limit where the model is capable of predicting within +- one hour for the whole route.

Std 0,00142 shows that the overall performance is good, not only a few very good performing and couple bad performing as with the generic model.

5.2 Navigation in the archipelago

Models difficulties to predict ETA within the archipelago

6 Discussion

Discussion about the results and validity future work

7 Conclusion

The possibility to use historical AIS data, example HELCOM dataset, to train a model on predicting the ETA for ports that have good coverage of vessels inbound or some amount of routes coming to the port

References

[1] F. Goerlandt, H. Goite, O. A. Valdez Banda, A. Höglund, P. Ahonen-Rainio, and M. Lensu, "An analysis of wintertime navigational accidents in the northern baltic sea," *Safety Science*, vol. 92, pp. 66–84, 2017, ISSN: 0925-7535. DOI: https://doi.org/10.1016/j.ssci.2016.09.011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925753516302193.

[2] Suomen Varustamot Ry, *Merenkulun avainluvut*, 2022. [Online]. Available: https://shipowners.fi/kilpailukyky/merenkulun-avainluvut/ (visited on 04/01/2022).

[3] International Maritime Organization, *Revised guidelines for the onboard operational use of shipborne automatic identification system*, 2015. [Online]. Available: https://www.imo.org/en/OurWork/Safety/Pages/AIS.aspx (visited on 12/14/2021).

[4] J. M. Mou, C. van der Tak, and H. Ligteringen, "Study on collision avoidance in busy waterways by using ais data," *Ocean Engineering*, vol. 37, no. 5, pp. 483–490, 2010, ISSN: 0029-8018. DOI: https://doi.org/10.1016/j.oceaneng.2010.01.012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S002980181000020X.

[5] J. Montewka, F. Goerlandt, P. Kujala, and M. Lensu, "Towards probabilistic models for the prediction of a ship performance in dynamic ice," *Cold Regions Science and Technology*, vol. 112, pp. 14–28, 2015, ISSN: 0165-232X. DOI: https://doi.org/10.1016/j.coldregions.2014.12.009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0165232X14002262.

[6] C. Jahn and T. Scheidweiler, "Port call optimization by estimating ships' time of arrival," in *Dynamics in Logistics*, M. Freitag, H. Kotzab, and J. Pannek, Eds., Springer International Publishing, 2018, pp. 172–177. DOI: 10.1007/978-3-319-74225-0.

[7] A. Alessandrini, F. Mazzarella, and M. Vespe, "Estimated time of arrival using historical vessel tracking data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 1, pp. 7–15, 2019. DOI: 10.1109/TITS.2017.2789279.

[8] S. El Mekkaoui, L. Benabbou, and A. Berrado, "Predicting ships estimated time of arrival based on ais data," in *Proceedings of the 13th International Conference on Intelligent Systems: Theories and Applications*, Association for Computing Machinery, 2020, ISBN: 9781450377331. DOI: 10.1145/3419604.3419768.

[9] A. Veenstra and R. Harmelink, "On the quality of ship arrival predictions," *Maritime Economics & Logistics*, vol. 23, no. 4, 655–673, 2021. DOI: 10.1057/s41278-021-00187-6.

[10] Trenz AG. (2021). Real-ETA, [Online]. Available: https://www.real-eta.com/ (visited on 12/14/2021).

[11] "Commission Directive 2011/15/EU of 23 February 2011 amending Directive 2002/59/EC of the European Parliament and of the Council establishing a Community vessel traffic monitoring and information system," 2011. [Online]. Available: https://eur-lex.europa.eu/eli/dir/2011/15/oj.

[12] A. Felski and K. Jaskolski, "The integrity of information received by means of ais during anti-collision manoeuvring," *International Journal on Marine Navigation and Safety of Sea Transportation*, vol. Vol. 7, No. 1, pp. 95–100, 2013. DOI: 10.12716/1001.07.01.12.

[13] "Directive 2009/16/EC of the European Parliament and of the Council of 23 April 2009 on port State control," 2009. [Online]. Available: http://data.europa.eu/eli/dir/2009/16/oj.

[14] G. Fancello, P. Claudia, M. Pisano, P. Serra, P. Zuddas, and P. Fadda, "Prediction of arrival times and human resources allocation for container terminal," *Maritime Economics & Logistics*, vol. 13, pp. 142–173, 2011. DOI: 10.1057/mel.2011.3.

[15] A. Harati Mokhtari, A. Wall, P. Brooks, and J. Wang, "Automatic Identification System (AIS): A Human Factors Approach," 2008.

[16] HELCOM, *Convention on the Protection of the Marine Environment of the Baltic Sea*, 2014. [Online]. Available: https://helcom.fi/about\-us/convention/ (visited on 03/29/2022).

[17] *Total transports in the Port of Naantali over 8 million tonnes in 2020*, 2021. [Online]. Available: https://portofnaantali.fi/en/press-release/total-transports-in-the-port-of-naantali-over-8-million-tonnes-in-2020/ (visited on 03/23/2022).

[18] Traficom, *Vesikuljetusten Kuljetusmäärät*, 2021. [Online]. Available: https://tieto.traficom.fi/fi/tilastot/vesikuljetusten-kuljetusmaarat (visited on 04/01/2022).

[19] Y. Wu and J. Feng, "Development and application of artificial neural network," *Wireless Personal Communications*, vol. 102, no. 2, 1645–1656, 2017. DOI: 10.1007/s11277-017-5224-x.

[20] K. Gurney, *An Introduction to Neural Networks*. USA: Taylor & Francis, Inc., 1997, ISBN: 1857286731.

[21] R. DiPietro and G. D. Hager, "Chapter 21 - deep learning: Rnns and lstm," in *Handbook of Medical Image Computing and Computer Assisted Intervention*, ser. The Elsevier and MICCAI Society Book Series, S. K. Zhou, D. Rueckert, and G. Fichtinger, Eds., Academic Press, 2020, pp. 503–519, ISBN: 978-0-12-816176-0. DOI: https://doi.org/10.1016/B978-0-12-816176-0.00026-0. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128161760000260.

[22] T. K. Gupta and K. Raza, "Chapter 7 - optimization of ann architecture: A review on nature-inspired techniques," in *Machine Learning in Bio-Signal Analysis and Diagnostic Imaging*, N. Dey, S. Borra, A. S. Ashour, and F. Shi, Eds., Academic Press, 2019, pp. 159–182, ISBN: 978-0-12-816086-2. DOI: https://doi.org/10.1016/B978-0-12-816086-2.00007-2. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128160862000072.

[23] B. Kumaraswamy, "6 - neural networks for data classification," in *Artificial Intelligence in Data Mining*, D. Binu and B. Rajakumar, Eds., Academic Press, 2021, pp. 109–131, ISBN: 978-0-12-820601-0. DOI: https://doi.org/10.1016/B978-0-12-820601-0.00011-2. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128206010000112.

[24] W. Di, A. Bhardwaj, and J. Wei, "Deep learning essentials: Your hands-on guide to the fundamentals of deep learning and neural network modeling," in. Packt, 2018, 81–90.

[25] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, 1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735.

[26] C. Olah, *Understanding LSTM networks*, 2015. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (visited on 04/26/2022).

[27] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, *TensorFlow: Large-scale machine learning on heterogeneous sys-*

tems, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/.

[28] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. DOI: 10.1038/s41586-020-2649-2. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2.

[29] J. Reback, jbrockmendel, W. McKinney, J. V. den Bossche, T. Augspurger, P. Cloud, S. Hawkins, M. Roeschke, gfyoung, Sinhrks, A. Klein, P. Hoefler, T. Petersen, J. Tratner, C. She, W. Ayd, S. Naveh, J. Darbyshire, M. Garcia, R. Shadrach, J. Schendel, A. Hayden, D. Saxton, M. E. Gorelli, F. Li, M. Zeitlin, V. Jancauskas, A. McMaster, P. Battiston, and S. Seabold, *Pandas-dev/pandas: Pandas 1.4.1*, version v1.4.1, Feb. 2022. DOI: 10.5281/zenodo.6053272. [Online]. Available: https://doi.org/10.5281/zenodo.6053272.

[30] K. Jordahl, J. V. den Bossche, M. Fleischmann, J. McBride, J. Wasserman, A. G. Badaracco, J. Gerard, A. D. Snow, J. Tratner, M. Perry, C. Farmer, G. A. Hjelle, M. Cochran, S. Gillies, L. Culbertson, M. Bartos, B. Ward, G. Caria, M. Taves, N. Eubank, sangarshanan, J. Flavin, M. Richards, S. Rey, maxalbert, A. Bilogur, C. Ren, D. Arribas-Bel, D. Mesejo-León, and L. Wasser, *Geopandas/geopandas: V0.10.2*, version v0.10.2, Oct. 2021. DOI: 10.5281/zenodo.5573592. [Online]. Available: https://doi.org/10.5281/zenodo.5573592.

[31] Wiki.GIS.com, *Decimal degrees*, 2011. [Online]. Available: http://wiki.gis.com/wiki/index.php/Decimal_degrees (visited on 04/04/2022).

[32] D. Choi, C. J. Shallue, Z. Nado, J. Lee, C. J. Maddison, and G. E. Dahl, *On empirical comparisons of optimizers for deep learning*, 2019. DOI: 10.48550/ARXIV.1910.05446. [Online]. Available: https://arxiv.org/abs/1910.05446.