Maritime Anomaly Detection using Autoencoders and OPTICS-OF

Henri Iltanen

Master's thesis
UNIVERSITY OF HELSINKI
Department of Computer Science

Helsinki, October 8, 2020

Acknowledgements

Writing this thesis has been a learning journey and a huge undertaking. It has not always been easy to find strength and time after a workday to do the research, but in the end, it was worth it and I'm both proud and grateful having finished it. Here are some of the acknowledgements which I want to share since, without the help of these people, this thesis would not have been possible:

I want to thank both of my thesis instructors for guidance and for sharing their expertise.

I want to thank Awake.AI Oy for providing the data, and a special thanks to my co-workers at Awake for interesting discussions, helping with getting the data and for pointing out the opportunity for the thesis.

I want to thank all my classmates and teachers for the past years at the university and also on their observations about the thesis which led to new insight.

I want to thank my parents, little sister and friends for their continuous support throughout the years.

Most of all, I want to thank my amazing and loving wife for her continuous support and encouragement and for putting up with me through the long evenings of working on this project.

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Faculty of Science	Department of Computer Science

Tekijä — Författare — Author

Henri Iltanen

Työn nimi — Arbetets titel — Title

Maritime Anomaly Detection using Autoencoders and OPTICS-OF

Oppiaine — Läroämne — Subject

Computer Science

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Master's thesis	October 8, 2020	88

Tiivistelmä — Referat — Abstract

Anomaly detection is an important task in many domains such as maritime where it is used to detect, for example, unsafe, unexpected or criminal behaviour. This thesis studies the use of deep autoencoders for anomaly detection on high dimensional data in an unsupervised manner. The study is performed on a benchmark data set and a real-life AIS (*Automatic Tracking System*) data set containing actual ship trajectories. The ships' trajectories in the AIS data set are a form of time-series data, and therefore recurrent layers are used in an autoencoder to allow the model to capture temporal dependencies in the data.

An autoencoder is a neural network architecture where an encoder network produces an encoding and decoder network takes the encoding intending to produce the original input. An encoding is a compressed fixed-sized vector presentation of the original input. Since the encoding is used by the decoder to construct the original input, the model learns during the training process to store essential information of the input sequence to the encoding.

Autoencoders can be used to detect anomalies using reconstruction error by assuming that a trained autoencoder is able to reconstruct non-anomalius data points more accurately than anomalous data points, and therefore data points with high reconstruction error can be considered anomalies. In addition to reconstruction error, the autoencoders produce encodings. The research of this thesis studies the possibility of calculating an outlier score for the encodings and combining the score with resconstruction error to form a combined outlier score. OPTICS-OF (*Ordering Points to Identify the ClusteringStructure with Outlier Factors*) is a density based anomaly detection technique which can be used to calculate outlier scores for the encodings. The outlier score of OPTICS-OF for a data point is based on how isolated it is within its neighbourhood.

The proposed method is evaluated on a benchmark Musk data set for which anomalies are known. A data set with labelled anomalies provides a setting for analyzing the performance of the method and its properties. The method is then put to the test on the AIS data set where it is used to find new anomalies in the data set from two derived distinct feature sets. The AIS data set contains one known anomaly which is presented both as an example of a maritime anomaly and for which more detailed analysis of the produced outlier scores are presented.

The results of the study show potential for the proposed combined score method, and the analysis identifies multiple areas for further research. Deep autoencoders are successfully used to find new anomalies from the AIS data set which show actual behaviour deviating from normal ship movement.

ACM Computing Classification System (CCS):
Machine learning

Avainsanat — Nyckelord — Keywords

Anomaly Detection, Deep Learning, Autoencoder, OPTICS-OF, Reconstruction Error, Maritime, AIS

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

1 Introduction **1**
1.1 Anomaly Detection and Maritime Traffic 1
1.2 Goals of the Research 2
1.3 Structure of the Thesis 3

2 Machine Learning **4**
2.1 Types of Learning and Problems 4
2.2 Machine Learning Model 5
2.3 Loss Function 7
2.4 Training Loop 8
2.5 Performance evaluation 9
2.6 Sequence Prediction Problems 11

3 Neural Networks **13**
3.1 Artificial neuron 13
3.2 Activation functions 15
3.3 Perceptron network 17
3.4 Feed-Forward Neural Network 19
3.5 Recurrent neural network 22
3.6 Long Short Term Memory Networks 24

4 Autoencoder Architecture **27**
4.1 Encoder - Decoder Structure and an Autoencoder . . . 27
4.2 Sparse Autoencoder 28

5 Anomaly Detection using an Autoencoder and Density Based Outlier Scores **30**
5.1 Anomaly Detection 30
5.2 Autoencoding as a Model-Based Technique 31
5.3 Density Based Technique for Analysing of Encodings . . 31
5.4 OPTICS-OF Density Based Technique 32
5.5 Combining Outlier Scores 36

6 Overview of Data and Feature Selection **39**
6.1 Musk Anomaly Detection Benchmark Data Set 39
6.2 Maritime Anomaly Detection using AIS Data 40
6.3 Cargo Ship Engine Failure as Known Anomaly 44

7 Methodology **46**
7.1 Research design 46
7.2 Normalization, Train - Test Data Split, Loss Function and Hyperparameters 47

7.3 Autoencoder with a Deep Encoder and a Shallow Decoder 49
7.4 Machine Learning Model Implementations 50

8 Results and Discussion 53
8.1 Musk Data Set 53
8.2 AIS Data Set 59

9 Conclusions 67

A Musk Data Numeric Results 69

B AIS Data Location Wise Features Numeric Results 76

C AIS Data Location Agnostic Features Numeric Results 80

1 Introduction

In this thesis, a novel anomaly detection method based on deep learning for high dimensional data is introduced. This introduction provides a high-level survey of anomaly detection and its use in maritime traffic. Then the goals of the research are described, and finally, the structure of the thesis is presented.

1.1 Anomaly Detection and Maritime Traffic

Anomaly detection is a problem which belongs under a Data Mining umbrella. Data mining is a process of discovering interesting and useful information from the data. The amount of data available for analysis has been growing rapidly[Pan14]. In many cases, the data points which deviate from the majority of the data points are the most interesting [BKNS99]. These data points can stem from abnormal behaviour; for example, in the financial domain, abnormal data points could indicate a fraud and in computer systems anomalous behaviour can indicate an intrusion to the system [Pan14]. The process of identifying abnormal data points, also called outliers or anomalies in a data set is called *anomaly detection.*

In maritime domain anomaly detection is referred to as Maritime Anomaly Detection [Bom06]. Detecting unsafe and suspicious activity such as drug smuggling, people trafficking, drunk sailing or piracy are examples of tasks of Maritime Anomaly Detection [Kow12]. As such, it has been identified as a critical part of increasing situational awareness in maritime traffic [Bom06]. AIS (*Automatic Identification System*) data is the primary method used by various maritime tracking services [PVD$^+$17]. All passenger ships and ships with gross tonnage 300 or more are required to have AIS transmitter on a ship. The AIS transmitter sends data of the ships location and additional metadata. The AIS data has been used to detect anomalies in ship traffic for example in [PVD$^+$17].

1.2 Goals of the Research

In this thesis, a data mining method taking advantage of deep learning and neural networks for detecting anomalies [Agg17] in an unlabeled data set is evaluated on two data sets. The proposed method is a combination and an extension of the methods presented in [PVD+17] and [YZZ+18]. To the best of of my knowledge the proposed method is new and has not been tried before. The goal of the research is to evaluate the performance of the proposed method and its components. In the proposed method, an autoencoder neural network is used to reduce the dimensionality of data and form compressed fixed size presentations of data points called encodings. An outlier score is calculated for the encodings using an OPTICS-OF algorithm [BKNS99]. The quality of the encodings is tied to reconstruction error. The goal is to evaluate the possibility of combining reconstruction error and OPTICS-OF analysis of encodings to form a combined outlier score. The robustness of the method is evaluated on a benchmark Musk data set and finally is used in practice to detect anomalies in a maritime domain on an AIS data set.

The AIS data set contains ship trajectories collected between 30.10.2019 and 14.01.2020 at the Gulf of Bothnia. The focus of the research is to find anomalies in commercial ship traffic and therefore the research limits itself to cargo and tanker ships which should follow shipping routes. A heat map of the AIS data set is presented in Figure 1 (left). The AIS data contains one known anomaly presented in Figure 1 (right), which is used for detailed analysis and as an example of an abnormal ship behaviour. The goal is to see which kind anomalies deep autoencoder can identify from the AIS data set using the proposed method.

The results of the research indicate potential for the proposed method and show that it can be successfully used to identify anomalies in the AIS data set. The results on the benchmark data set highlight behaviour of both OPTICS-OF and reconstruction error, and similar behaviour is also visible on the AIS data set. The correlations between anomaly detection efficiency, model complexity and training data quality are presented. The shortcomings of the proposed method are identified and potential solutions outlined for further research.

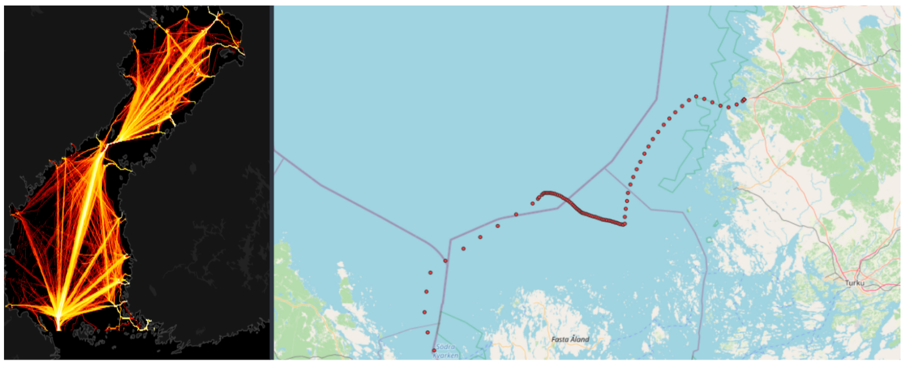

Figure 1: A heat map of the AIS data used in the research is presented at the left of the image. On the right, an example anomalous trajectory is displayed.

1.3 Structure of the Thesis

This thesis starts with an overview of machine learning and sequence prediction problems. It then continues to introduce neural networks and focuses on recurrent neural networks and their variant LSTM (*Long Short Term Memory*) networks [GBC16], which are suitable for time series analysis and sequence prediction. Thus LSTM's possess abilities which are required in the analysis of the AIS data. Chapter 4 presents autoencoder architecture. The details of the evaluated anomaly detection method are presented in Chapter 5. The methodology of the research is presented in Chapter 7. After which results and conclusions follow.

2 Machine Learning

In machine learning, a computational model improves its performance by experience. In his book *"Machine Learning The Art and Science of Algorithms that Make Sense of Data"* Peter Flach [PeF12] gives the following definition for machine learning *"Machine learning is the systematic study of algorithms and systems that improve their knowledge or performance with experience"*. In machine learning the goal is to learn structure and relationships in the data which can be used to understand the data and possibly make predictions based on it. Machine learning has benefited from the vast growth of available computing power and data. Machine learning has been used to solve problems in various fields, including medicine, astronomy, and engineering [GaJ15]. In this chapter, general terminology, the process of training and evaluating machine learning models is introduced.

2.1 Types of Learning and Problems

At the time of writing, there are three fields that machine learning can be roughly categorized to: supervised learning, unsupervised learning, and reinforcement learning [PeN10]. The categorization is based on the type of data used for learning and goals of tasks being solved.

Supervised Learning: In supervised learning, the model is given input-output pairs, and the task is to learn a function from input to output [PeN10]. For example supervised learning data could contain input features for a house price such as the number of rooms, number of floors and floor area and then the house price acts as the output variable [GaJ15].

Unsupervised Learning: Unsupervised learning systems learn from data which does not have any labels. When there are no labels in the data the best the learning system can do is to learn patterns in the input data. Extracting customer groups from customer data is an example of unsupervised learning [PeN10].

Reinforcement Learning: In reinforcement learning an agent learns by interacting with an environment. In reinforcement learning the data comes in the form of observations and rewards from the environment. Reinforcement learning can be used for example to teach robots to walk where they observe the world and their body through sensor measurements. The reward in the example can be a function which includes, for example, the stress on joints, walking speed and how upright the robot is walking [RiS18].

In this thesis, the research being conducted is an unsupervised learning scenario, since the data being used is not labelled from the perspective of the model. However, the analysis is going to be using supervised learning techniques to construct and train a model. As the research in this thesis illustrates the categories of machine learning have many similarities and dependencies in terms of techniques and methods, and often lines between categories can get blurred. In this thesis, the focus is both on unsupervised and supervised learning since methods from both categories are heavily present in the research.

Supervised learning prediction problems can be roughly divided into two categories according to the type of output value: regression and classification tasks. In regression tasks, the output value is continuous. An example of this is predicting the price of a stock on the market. In a classification task, the goal is to predict a category for the input. Therefore in classification tasks, the output value is discrete. An example of a classification task is predicting a breed of a dog in an image. There are many dog breeds, and so there are multiple categories, but the output value is discrete, not continuous. Classification problems with multiple categories are referred to as multiclass-classification problems. In this thesis the problems are regression problems where the output values are continuous.

2.2 Machine Learning Model

In machine learning, the goal is to estimate a function f, which describes relationships and structure in the data.

Definition 1. Let us assume there is a relationship between a value vector of interest Y and a feature vector X, which can be presented in the following form.

$$Y = f(X) + \epsilon$$

Function f is some unknown function and ϵ is the general error term. [GaJ15]

The value of interest Y is the sum of a function f with an input X and the general error term ϵ. In all real-world situations, most often the feature vector X cannot capture everything needed to predict the value of interest Y. The general error term ϵ contains this inaccuracy. The general error term also includes error caused by measurement inaccuracy in case there is some. Machine learning methods can not reduce the error included in the general error. The function f describes the relationship between feature vector X and the value vector Y. With machine learning methods, one can produce an estimate of f. By choosing and training an appropriate model, one can reduce the error by getting a better estimate $\hat{f}$ of f.

Machine learning can be used both to understand the data and to make predictions. When the goal is to gain a deeper understanding of the data and not necessarily to make the best possible predictions, requirements for the method are different than in a pure prediction task. When the goal is solely to make good predictions machine learning model can be treated as a black box which produces predictions. When, however, one wishes to get insight into the relationships in the data, the model has to describe the structure it has discovered understandably. In many cases, the simpler the model, the easier it is to interpret and the other way around the more complex model, the harder it is interpreted. On the other hand, as model complexity grows the ability for it to estimate function f increases and therefore, it can make better predictions unless it becomes subject to *overfitting*; a phenomenon presented in the next section.

2.3 Loss Function

In training machine learning models, the goal is to find a better estimate $\hat{f}$ of the function f. In order to get a better estimate, one has to be able to measure the difference between the prediction produced by $\hat{f}$ and the correct value Y. The difference is called *loss*, and it is calculated using a loss function. A loss function is referred to in numerous sources with various terms in addition to loss like error - and cost function. In this thesis, the term loss function is used.

As loss function measures the loss in the predictions of the model, it is central for the learning process and has to be selected appropriately for a task at hand. A loss function takes a vector of predictions and the corresponding correct values as an input. The output is a single scalar which describes the combined loss for all provided inputs. Let us examine loss functions for classification and regression tasks. The most intuitive way of calculating the error in a classification task is to calculate the amount of miss-classified data points. A loss function which calculates the amount of miss-classified data points is called an absolute loss function. The corresponding absolute loss function for a regression task is a function which computes the distance between the prediction and the true values.

The equation for calculating an absolute loss. The d is a vector of correct values and y presents a vector of predicted values. For a classification task we have

$$\frac{1}{n}\sum_{i=0}^{n-1}\begin{cases}1, & \text{if } d_i \neq y_i \\ 0, & \text{otherwise }, \end{cases} \tag{1}$$

and for a regression task

$$\frac{1}{n}\sum_{i=0}^{n-1}|d_i - y_i|. \tag{2}$$

The values are accumulated by calculating an average.

There are many properties which are desirable for a loss function. These include: robust to outliers in the data, continuous and insensitive to noise in the data. Different loss functions have a different set of properties. There are many loss functions which include, for example, mean squared error, log loss, and hinge loss. Mean Squared Error is a loss function for calculating loss for continuous values (see Definition 2). It takes an average squared absolute error. It penalizes predictions which are far from correct value comparatively more since the loss grows exponentially relative to the absolute error.

Definition 2. Mean Squared Error. The d is a vector of correct values and y is a vector of predictions.

$$\frac{1}{n}\sum_{i=0}^{n}(d_i - y_i)^2$$

2.4 Training Loop

Once a loss function has been selected the process of training the model can be treated as an optimization problem. The goal is to find parameters θ for the model $\hat{f}$, which minimize the selected loss function. The set of all possible parameter selections is called a search space. The search space often contains multiple local minimums, and in many practical problems, the goal is to find an acceptable local minimum since finding the global minimum for a complex search space is extremely difficult. In order to avoid ending up to the same local minimum each training time, the parameters of the model are first given random values, so each time, different sections of the search space can be explored. There are also other ways to select initial parameter values which can be handcrafted for the chosen model architecture; an example of such parameter initialization technique is described in [HZRS15].

Exploring the search space for feasible parameter values can be performed using a variety of methods like *gradient descent* or *evolutionary algorithms.* In this thesis, the optimization of the model is performed using an *Adam optimizer* [KB14] which is a method which builds upon gradient descent optimization. The detailed description of gradient descent and its variants are not in the scope of this thesis, but a high-level description is presented next. Further information can be obtained from [KB14].

A *gradient* for a parameter w is defined as the partial derivative of the parameter in respect to the loss function. Optimization using gradient descent is simply taking steps to the direction of the steepest descent. Meaning the direction where the derivative of the loss function is lowest. The update step of gradient decent is defined in Definition 3. As the gradient descent uses the derivative, it requires the loss function to be partially differentiable. The length of each step in machine learning setting is often referred to as the learning rate. The too-high learning rate can lead to overstepping good solutions, and on the other hand, too small learning rate can lead to getting stuck to a suboptimal local minimum or cause slow convergence. Learning rate can be a static scalar value, or it can be adjusted during the training, for example, by starting with a higher learning rate and gradually lowering it as a function of taken steps.

Definition 3. Iteration step for updating parameter w with gradient descent. w_{new} is the updated parameter value, w_{old} is the old parameter value and α is the learning rate. The $\frac{\partial E}{\partial w}$ is the partial derivative of the loss function in respect to the parameter w.

$$w_{new} = w_{old} - \alpha \frac{\partial E}{\partial w}$$

In the training process, a data point of the data set is passed to the model, which then outputs a prediction based on it. The prediction is compared to the label of the data point, and an error is calculated using a loss function. A gradient descent iteration step is performed on the parameters of the model. The number of data points which are processed between parameter updates is called batch size. Batch size is a hyperparameter which with the learning rate has a significant effect on the results of the training. [HHS17] [SKL17]

2.5 Performance evaluation

When evaluating the prediction accuracy of a supervised learning model, one needs to consider does the model generalize well to unseen data. Often it is possible to fit a complex enough model almost perfectly to the data used for the training and as a result, get a model which generalizes poorly on unseen data. This phenomenon is called *overfitting*.

In machine learning, there is a trade-off between bias and variance. In machine learning context variance describes the models sensitivity to noise and individual data points in the training data. Bias is the simplified assumptions which the model makes about the patterns in the data. The variance increases with model complexity, and bias decreases. Ideally, one would want to have a model with a low bias and a low variance. High variance leads to overfitting, and high bias leads to underfitting. *Underfitting* is the opposite of overfitting and means that the model does not have enough expressiveness to describe the relationships in the data to make good predictions.

To test performance of a model, one has to cut the data set into at least two parts: the test set and the training set. Data in the test set is not used during the training of the model but is used to evaluate performance only after the training is completed. Correspondingly data in the training set is only used to train the model, and good accuracy on the training set cannot be used to state that the model has a good performance. Plotting the losses during training for test and training set shows that initially, loss decreases for data points on both sets, but at some point loss on the test set starts to increase while the loss on training set continues to decline. The point where the loss on test set starts to increase marks the point where the model starts to overfit the data. The overfitting is visualized in Figure 2.

In addition to training and test sets, it is beneficial to have a validation set. The validation set is not used for training but is used for tuning the various hyperparameters of the model, for example, neural network architecture, batch size and learning rate (see Chapter 3). Since the hyperparameters are fine-tuned for best accuracy on the validation set only the test set which has not been used building the model can be used to determine how well the model performs on unseen data.

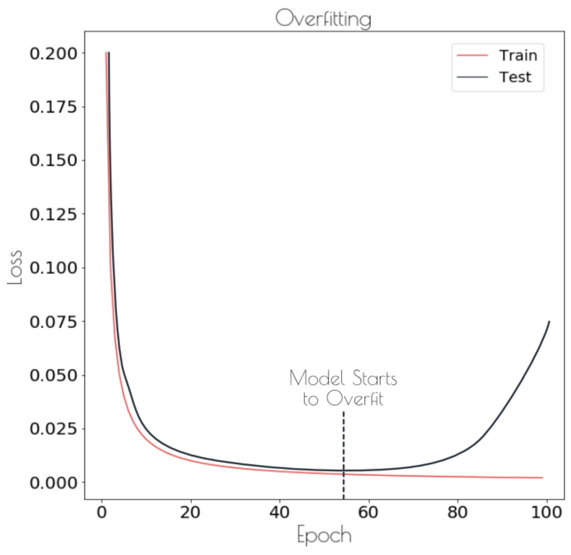

Figure 2: Training progress for a model showing how the model eventually starts to overfit the training data.

2.6 Sequence Prediction Problems

Machine learning models can be used to discover patterns in various kinds of data, for example, images and text. In this thesis, the AIS data is time series data which can be modelled as a set of sequences. This section introduces sequences in a machine learning and data mining context.

In traditional supervised machine learning problem, the data is a set of observations where each observation is a feature vector. A sequence prediction differentiates from the traditional supervised problem in a few fundamental ways. In a sequence prediction problem, each observation is a list of feature vectors where the order is essential. Example of such a sequence is for example text and a sequence of frames in a video recording. In the case of text, for example, the meaning of a phrase can change completely if the order of words is altered. For example in case of movie reviews two reviews *"The movie was entertaining, not boring at all."* and *"The movie was boring, not entertaining at all."* even though they contain same words. A data set for a sequence prediction problem is a set of sequences. Another typical speciality for sequence prediction problems is that they must deal with various lengths of sequences. Considering text data as an example, the sentences in text consist of various amounts of words.

A sequence prediction problem can be formalized as follows: a sequence to modelling prediction has a set of S sequences as data:

$$S = X_0, X_1, ..., X_m$$

where each sequence X_i contains feature vectors for all n_j time steps

$$X_i = (x_{i0}, x_{i1}, ..., x_{in_j})$$

All feature vectors have the same dimensions, but lengths n_j may vary between sequences. The output of a sequence prediction model can be a scalar, a fixed size vector or a variable-sized sequence. Sequence prediction problems can be divided into different categories according to the type of output:

- Predicting scalar values for a sequence (example: predicting the next element in a sequence)
- Predicting a label for a sequence (example: prediction whether a review is positive or negative).
- Generating a new sequence from a sequence (example: music generation).
- Predicting a Sequence from a sequence (example: machine translation).

Sequence prediction problems require sophisticated models to efficiently discover patterns in sequential data. Models requirements also vary on the type of output. This research focuses on the sequence to sequence prediction and in the next chapter, a model suitable for sequence prediction is presented.

3 Neural Networks

A neural network is a machine learning model which was inspired by the structure of biological neural networks [PeN10]. Neural networks have also gotten many terms from their biological inspirations: nodes in a neural network are often called neurons, and there are memory units in recurrent neural networks [Bra16]. Artificial neural networks have a history starting from 1943 when Warren McCulloch and Walter Pitts developed the first model for artificial neuron [PeN10]. After initial research, the interest in neural networks has experienced highs and lows. During mid-80's neural networks were popular in the artificial intelligence research community but on the 1990s lost some of their appeal. After 2010 neural networks became again very popular under a new name *deep learning* [Bra16]. Deep learning's success was enabled by increased computational capacity, availability of large data sets and improved training methods. Deep learning has achieved unprecedented success in many machine learning problems [GMfMPIfMitS14]. In this chapter, neural networks are introduced starting from a single artificial neuron and gradually moving from a single layer neural network, the perceptron network to more complex networks ending at a deep recurrent neural network.

3.1 Artificial neuron

A neural network consists of layers of nodes and edges between nodes on separate layers. Each layer can have multiple artificial neurons which in this thesis are referred as nodes. [PeN10] In this chapter, the building block of a neural network, the artificial neuron is introduced. Artificial neurons loosely mimic neurons of the human brain.

An artificial neuron has three parts: input function, activation function, and an activation. The artificial neuron gets as an input two vectors: values x and weights w, which both have a length i. The input function combines the values and weights into a single value which is fed to the activation function. The most popular input function is a weighted sum function, but in theory, other kinds of input functions could be used.

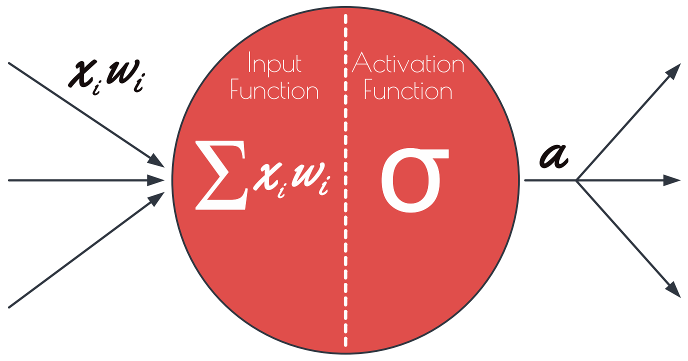

Figure 3: In the figure, the input values x_i and weights w_i come from the left links. The input function weighted sum is used to combine i values and weights into a single value. Then the value is passed to an activation function which outputs the activation a of the artificial neuron which is then outputted to the right links. [PeN10]

The activation function is applied to the result of the input function. Activation function can be used to control the output of the artificial neuron. The activation function mimics biological neurons ability to decide whether or not to produce an activation. Activation functions have a central role in the expressive power of the neural networks since non-linearity is introduced through non-linear activation functions. A neural network with only linear activation functions is unable to approximate non-linear relationships in the data. There are many different commonly used activation functions, the ones used in this thesis are presented in Section 3.2.

The final output of the artificial neuron is the output of the activation function. This value is called an activation. Now we can write a simple mathematical model for an artificial neuron which is presented in Definition 4.

Definition 4. A mathematical model for an artificial neuron using a weighted sum as an input function. a is the activation of the artificial neuron, x_i is the ith the input value and w_i is the weight of the ith input link. σ is an activation function which outputs the activation a.

$$a = \sigma(\sum_{i=0}^{n} x_i w_i)$$

3.2 Activation functions

Activation functions have different properties, and choosing a correct activation function is critical when designing neural networks. We are now going to examine the following activation functions step-, sigmoid-, tanh- and ReLu functions (functions are plotted in Figure 4).

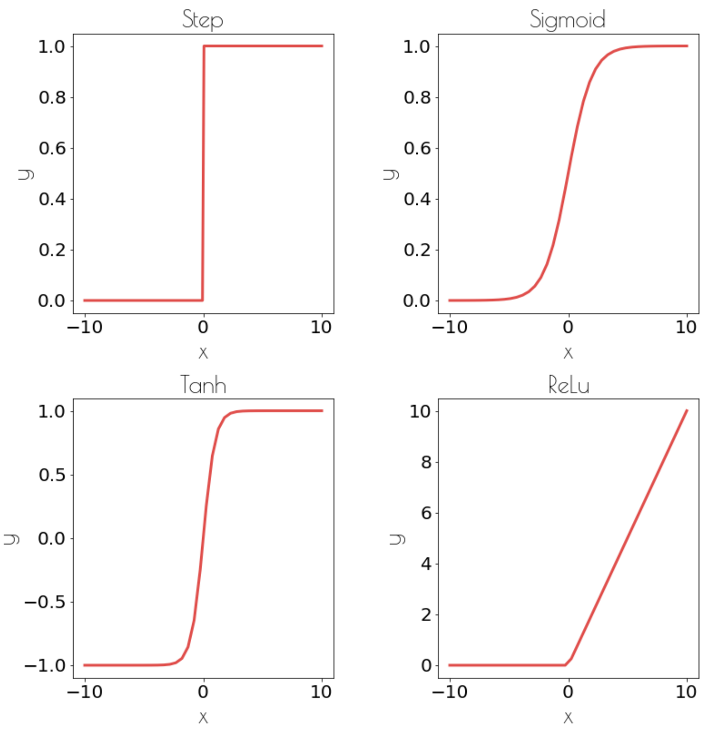

Figure 4: The figure shows plots of step-, sigmoid-, tanh- and ReLu activation functions.

The step activation function defined in Definition 5 can be used between the layers of a network and to output a boolean value from a network. The step activation function is used in the perceptron example presented in Section 3.3. The Relu activation function defined in Definition 6 is commonly used between layers of the network. It is also useful for outputting a positive scalar value from the networks output layer. ReLu adds non-linearity to the plain linear activation function $f(x) = x$ being a non-linear activation function gives ReLu an advantage and makes it a more powerful function approximator than linear alternatives. In the research of this thesis, ReLu was tested, but for some reason, a tanh activation function yielded better results. The tanh function defined in Definition 8 is a common choice for an activation function between LSTM layers.

Definition 5. The step activation function for an artificial neuron:

$$f(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases}$$

Definition 6. The ReLU activation function.

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases}$$

The sigmoid activation function is defined in Definition 7. The sigmoid activation function always outputs a value between zero and one. It is well suited for outputting confidence or probability as an output. Sigmoid also has an essential role in LSTM networks presented in Section 3.6 where the sigmoid function is used to decide which information to remember and which to forget.

Definition 7. The sigmoid activation function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Definition 8. The tanh activation function:

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

3.3 Perceptron network

Perceptron network is a feed-forward neural network with a single layer, meaning that all the inputs are directly connected to the outputs. [PeN10] Perceptron was first proposed by Frank Rosenblatt in 1958 [Ros58]. In this chapter, it is shown by an example how a perceptron network can be used to solve a simple classification problem.

The data of the example has two features x_0 and x_1. For each pair of features, there is a label y which is either one or zero. The data has four hundred data points in total generated from two normal distributions. The task is to find a linear-separation in the data which could be used to predict the label for future data points. In this example, we are not doing a model evaluation, but instead, the focus is on the process of training the perceptron network.

The input function is chosen to be the weighted sum. Since we want to output a value which is either one or zero, we are going to use the step function as an activation function (see Definition 5).

Since there is a single output, the perceptron network has one neuron where all edges of the input layer are connected (see Figure 5). The input features are x_0 and x_1, and also, a bias input with a value of one is added. The weights connecting the inputs to the output node are marked with a row matrix W, which has three values.

$$W = [w_0, w_1, w_2]$$

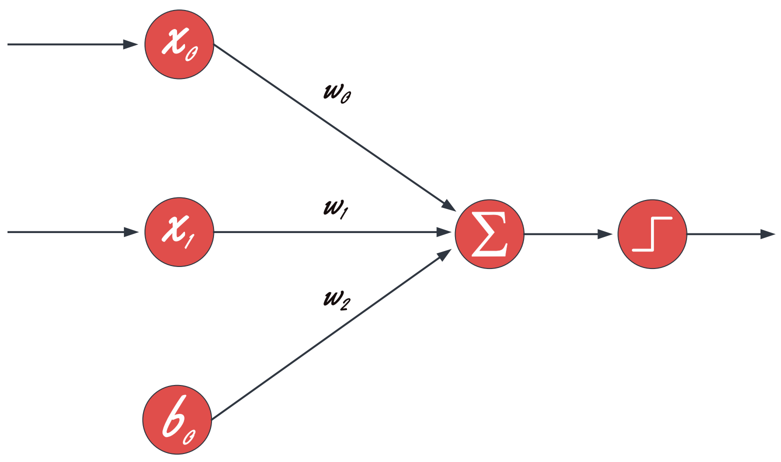

Figure 5: The image describes the architecture of the perceptron used in the example. The x_0 and x_1 are the inputs features, b_0 is the bias which is selected to have value one. The weights w_0, w_1 and w_2 are marked to the corresponding links. Weighted sum and step function are presented as separate nodes in the figure as details. These details are omitted in upcoming figures.

The output of a perceptron network for an input vector $\vec{x} = [x_0, x_1, 1]$ can be calculated using the Equation 3.

$$a = \begin{cases} 1 & \text{if } \vec{x} \cdot \vec{w} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$w_i = w_i + \alpha \cdot (d_j - y_j) \cdot x_{ij} \tag{4}$$

During the training of the perceptron, the weights w are updated using Equation 4. The subscript j is the index of a data point. α is the learning rate. The weights were initialized randomly. A learning rate of $\alpha = 0.001$ was used for training the perceptron for 100 epochs. Figure 6 visualizes the training results. In the figure, the two classes in the data set are visualized, and a decision boundary learned by the perceptron. By the epoch 100, the perceptron has discovered quite good decision boundary for separating the two classes. In order to solve more complex problems, neural networks with non-linearity and more layers are required. This is the topic of the next section.

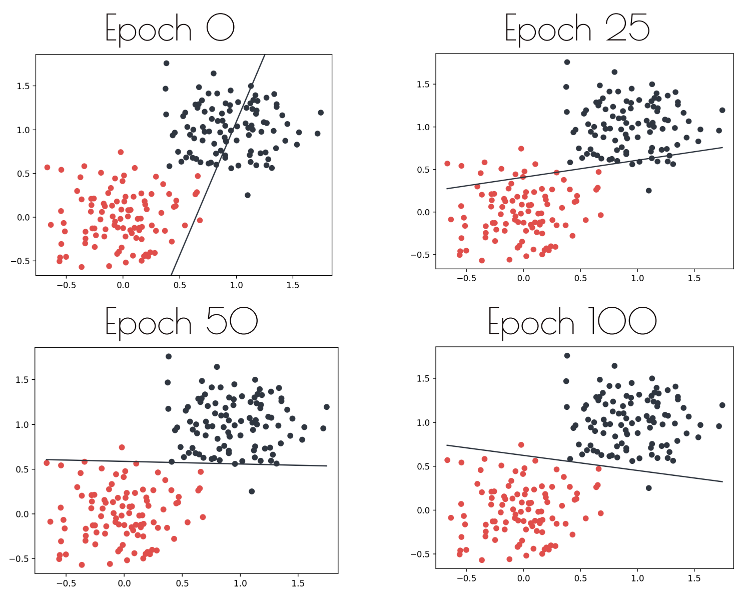

Figure 6: The images present the increase of accuracy for the perceptron network while it is trained.

3.4 Feed-Forward Neural Network

A feed-forward neural network is a multilayered perceptron consisting of an input layer, a number of hidden layers and an output layer. The nodes on each layer are connected to all nodes on the next layer. There are no other edges in a feed-forward neural network. In Figure 7, a feed-forward neural network with one hidden layer is presented.

The input layer has a node for each input of the network and a bias node. The bias node always has activation of one. Activation of the other nodes at the input layer is the input which was given to the network. The layers between the input layer and the output layer are called hidden layers. Each of the hidden layers has several nodes, an activation function and a bias node with a constant value of one. The output layer is the final layer of the network, and its activation is the output of the network. The output layer has nodes and an activation but no bias node.

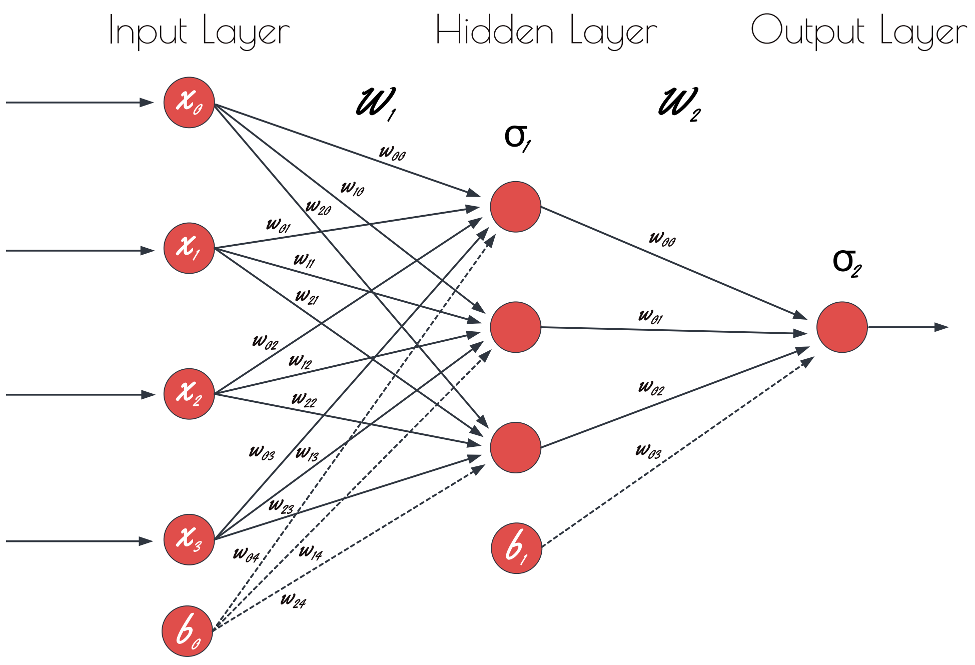

Figure 7: Feed-Forward Neural Network with an input layer, output layer and a single hidden layer. The σ present the activation functions and b the bias. There are two weight matrices W_1 and W_2. Individual weights of the weight matrices have also been marked to the figure to corresponding links.

Input and hidden layers are connected to the next layer by a set of edges. Each edge has a weight associated with it. The weights for each set of edges are stored in a matrix W_l where the index specifies the layer the edge is connected to. The individual weights are indexed w_{tofrom}. The weights of the network have been marked to the Figure 7. In the matrix W_l the row specifies which node at the target layer the edge with the weight is connected to and the column specifies from which node the edge is coming from on the previous layer. Hence the matrix W has dimensions $k \times i$ where k is the number of nodes on the target layer (not including the bias since there are no edges coming to bias node) and i is the number of nodes on the previous layer (including the bias node).

$$\begin{bmatrix} w_{00} & w_{01} & \cdots & w_{0k} \\ w_{10} & w_{11} & \cdots & w_{1k} \\ \vdots & \vdots & \ddots & \vdots \\ w_{i0} & w_{i1} & \cdots & w_{ik} \end{bmatrix}$$

The input for the neural network is presented as a vector x. The vector $x = [x_0, x_1, ..., x_k]$ becomes the activation of the input layer.

The process of calculating an output in a feed-forward neural network is called feed-forward. Calculating the output of a feed-forward neural network is merely calculating the activation for each layer and feeding a vector of the activations' as an input to the next layer until the output of the output layer has been calculated hence the name feed-forward. The process has been formulated in the Equation 5. [GBC16]

$$a_l = \sigma_l(W_l \times a_{l-1}) \tag{5}$$

To use a gradient-based optimization algorithm to train the neural network, the gradients of the loss function respect to the weights of the neural network need to be calculated for each forward pass. In practice, most neural network machine learning libraries, including Tensorflow used in the research of this thesis use automatic differentiation to calculate the gradients. The details of automatic differentiation are outside the scope of this thesis, but more details can be obtained from [PGC+17].

Once the gradients have been computed. The optimization can be performed using an optimization algorithm like stochastic gradient descent or one of its extensions such as Adam optimizer. The optimizer will update the weights based on the gradients and the learning rate to minimize the loss specified by the loss function. This process is called backpropagation. [GBC16]

Feed forward neural networks are well suited for tasks where there are fixed-sized inputs and outputs. They, however, are not well suited by themselves for sequence prediction problems. The feed-forward neural networks can only take a fixed-size vector as an input, but the length of sequences can vary. Methods like a sliding of a fixed sized window or padding and truncating inputs can be used to transform sequences into a form which they can be fed to a feed-forward network, but these approaches come with their own set of problems. For a sliding window, only elements within the window can be used to make a prediction if meaningful data falls outside the window the prediction might lose accuracy. If the size of the window is large, then only some set of parameters will see specific parts of the sequence and shifting the sequence causes the network to see input on parameters which have never seen it before. In other words, the learned information about some part of the sequence does not transfer if the part is encountered in another part of the window. In summary, if only few time steps are always required to make an accurate prediction feed-forward neural network might be a good choice, but in general it is not an ideal choice for a sequence prediction problem. [GBC16]

3.5 Recurrent neural network

Recurrent neural networks address the shortcoming of feed-forward neural networks being able to accept only fixed-size inputs and produce only fixed-size outputs and not being able to take into account previous inputs in the same sequence. By considering previous inputs in the sequence, recurrent neural networks are able to capture temporal dependencies which standard feed-forward neural networks are not able to capture. Recurrent neural networks ability to accept variable-length inputs and produce variable-length outputs and capture temporal dependencies make them ideal for sequence modelling problems. [GBC16]

Recurrent neural networks are neural networks which take a sequence as an input and then for each time step evaluate the neural network. A recurrent neural network can be thought as a neural network which has a loop allowing it to maintain a state. When evaluated, the loop is unwound through the time steps of a sequence as in Figure 8. The input for each time step consists of the features of the time step and hidden state of the recurrent neural network. Hidden state h is updated on each time step and propagated to the next time step. This way, the recurrent neural network is able to consider previous inputs

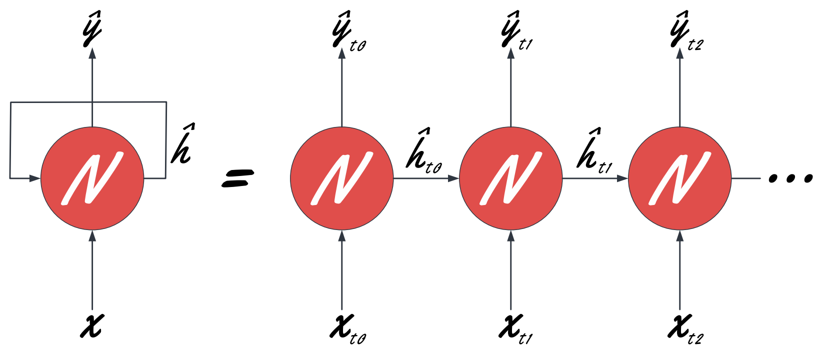

Figure 8: A recurrent neural network where output $\hat{y}$, input x and state h are presented. N indicates a neural network which is evaluated each time step. This could be, for example, feed-forward neural network or a convolutional neural network. Unfolding the loop in the recurrent neural network through time steps in a sequence is illustrated.

via the hidden state. The equation for calculating the hidden state at each time step is presented in Equation 6 [GBC16]

$$h_t = f(W_h \times h_{t-1} + W_x \times x_t) \tag{6}$$

The success of feed-forward neural networks and neural networks in general in approximating complex functions is attributed from the ability to stack multiple layers. When a neural network has multiple layers, each of the layers solves some part of the task, and the result is fed to the next layer, which then builds upon it [HS13].

In one sense, recurrent neural networks are by default multilayered since they are unfolded through time steps. In addition to unfolding through time steps, recurrent neural networks can also be stacked. By stacking recurrent neural networks, the network is able to consider the time series on multiple time scales. [HS13] Stacked recurrent neural networks have been shown to solve many complex problems with long temporary dependencies.

Recurrent neural networks are trained using a method called backpropagation through time. In the algorithm, the gradients have to be calculated for each time step using the chain rule. When the sequences are long, the backpropagation requires a vast number of multiplications. This can lead to a problem called vanishing or exploding gradients. The vanishing exploding gradient problem means a situation where the gradients either become very small; hence the term vanish or become very large, i.e. explode. When gradients are very small, also the learning updates become minimal and cause the learning to stall. [GBC16]

On the contrary, huge gradients cause the learning steps to take leaps which often leads to overstepping and prevents closing in on a good solution. The vanishing-exploding gradient problem limits the recurrent neural networks ability to learn long temporal dependencies. As a solution, multiple variations and extensions of vanilla recurrent neural networks have been developed, one of which is introduced in the next section. [GBC16]

3.6 Long Short Term Memory Networks

Long Short Term Memory Networks LSTM networks reduce the exploding or vanishing gradient problem [GBC16]. LSTM has a cell state which controls the flow of information between time steps in addition to the hidden state. LSTM has gates which are used to modify the state and form an output. An overview of LSTM is presented in Figure 9.

The forget gate is used to forget irrelevant information from the cell state. The forget gate has its own weight matrix. The inputs of the forget gate are the previous hidden state h_{t-1} and the current input x_t. A sigmoid function is used to output a value between zero and one for each of the elements in the cell state. An element-wise multiplication is performed between the output of the forget gate and the cell state. Since the output of the forget gate is between zero and one for each element effectively the value of one means keep the information of the element in the cell state and zero means completely forget the information in the element of cell state. The formula for the forget gate is presented in Equation 7. [GBC16]

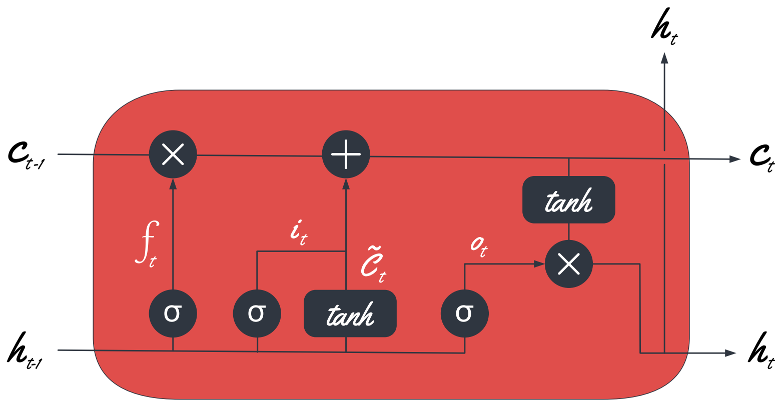

Figure 9: An illustration of a single long short term memory networks memory cell. Illustration contains forget-, update and output gates and variable symbols which correspond to the ones presented in the equations.

$$f_t = \sigma(W_f \times [h_{t-1}, x_t] + b_f) \tag{7}$$

The second operation in an LSTM cell is the update gate. The update gate identifies new information from the current time step and previous hidden state, which to include in the cell state. This is done in two parts: decide which values to update and generating candidate values for the update. As in forget gate first, a sigmoid function with a weight matrix and inputs are used to calculate a vector i_t where each value is between zero and one. The vector i_t is used to select values from the potential new candidates to include in the cell state. The candidate vector C_t also has its own weight matrix and uses previous hidden state and inputs to form a vector with the same dimensions as the cell state. A tanh function is used as a non-linear function from the output values. Then an element-wise multiplication is performed between the selection vector i_t and candidates C_t to select which new information is to be included in the cell state. Finally, the result of the multiplication is added to the cell state with vector addition. The process has been formalized in Equations 8 and 9. [GBC16]

$$i_t = \sigma(W_i \times [h_{t-1}, x_t] + b_i) \tag{8}$$

$$\widetilde{C_t} = \tanh(W_C \times [h_{t-1}, x_t] + b_C) \tag{9}$$

The forget and update gates define how the cell state is updated each time step. The update for the cell state on a time step has been formalized in Equation 10.

$$c_t = f_t * c_{t-1} + i_t * \widetilde{C_t} \tag{10}$$

The final output of an LSTM cell is the hidden state h_t. The calculation for h_t is presented in Equations 11 and 12. The calculation has the same gate structure as the forget and update gates. First, a sigmoid function is used to calculate a vector with values between zero and one which is used to select which values of the cell state are output at the time step. [GBC16]

$$o_t = \sigma(W_o \times [h_{t-1}, x_t] + b_o) \tag{11}$$

$$h_t = o_t * \tanh(c_t) \tag{12}$$

The way LSTM mitigates the vanishing or exploding gradient problem is evident by taking a closer look at the Equation 10 which specifies the calculation for updating the cell state at each time step. The updating of the cell state at each time step contains only element-wise multiplication and addition and no matrix multiplications, which was the case with vanilla recurrent neural networks. This allows gradients to have an undisturbed flow which significantly aids the training back-propagation process. For this reason, LSTM's are more effective at detecting long term temporal dependencies than vanilla recurrent neural networks. LSTM's can be stacked in the same manner as vanilla recurrent neural networks to form deep LSTM networks. In practice LSTMs have proved to be very succesfull in many applications. [GBC16]

4 Autoencoder Architecture

Nodes in a neural network can be organized to form many types of architectures. The architecture of a neural network is central for its performance and abilities. In this thesis, an autoencoder architecture is used since it has the ability to construct meaningful low dimensional presentations of data called *encodings*. An autoencoder is suitable for unsupervised learning setting. Autoencoder is trained in a supervised manner, but the output is the input itself, and therefore autoencoder is an unsupervised learning technique. Next autoencoder and its components are presented in more detail.

4.1 Encoder - Decoder Structure and an Autoencoder

Autoencoder utilizes an encoder-decoder structure. In an encoder-decoder structure, the encoder outputs a fixed-length vector presentation which is referred to as encoding. The encoding is used as an input to the decoder, which then constructs the final output of the network. Encoder decoder architecture is visualized in Figure 10. Encoder-decoder architecture can be used for example for solving sequence to sequence prediction problems like machine translation, where the encoder encodes the sentence from the original language to an encoding from which the decoder then constructs a corresponding sentence on the target language [CVMG$^+$14].

The name of the autoencoder comes from the idea that the encoder and decoder are trained together with the goal of reproducing the original input. Once autoencoder has been trained, the encodings present a fixed-length presentation of the input encapsulating some of the relevant information. Autoencoders can be used for tasks like dimensionality reduction, data compression and feature learning. [JZZX13]

In many sequence to sequence prediction problems, the inputs and outputs have varying sizes. The encoder-decoder architecture can be coupled with recurrent neural networks to construct a network which can consume variable-size inputs and produce variable size outputs. [CVMG$^+$14]

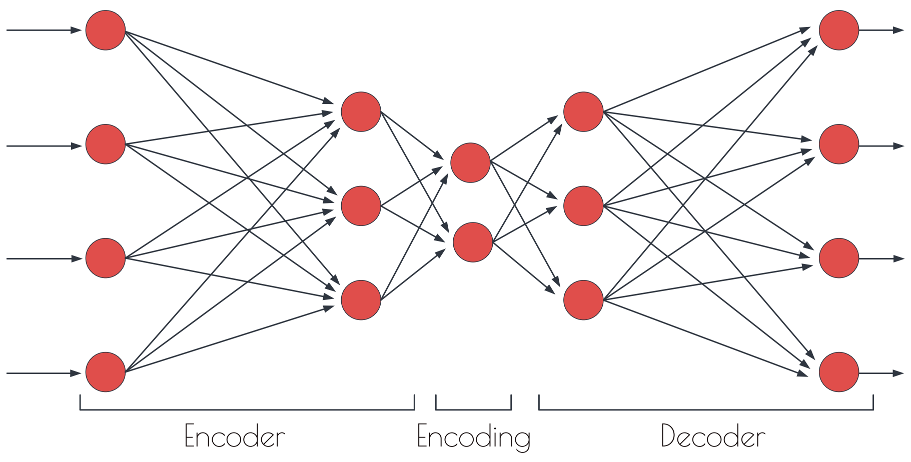

Figure 10: An encoder-decoder architecture and its parts encoder, encoding and decoder a presented on a high level. The illustration illustrates a simple feed-forward based architecture. Other variations can be implemented by replacing encoder and decoder with other types of neural networks.

4.2 Sparse Autoencoder

It has been observed that while using autoencoders for dimensionality reduction and feature learning sparsity can improve the quality of the produced encodings [LZW18]. Sparsity in an encoding means that a significant proportion of values in the encoding are zero or near zero. The goal of enforcing sparsity on an encoding of an autoencoder is to reduce the number of unused features and to improve the model's generalization [LZW18].

The architecture of an autoencoder can be used to guide the network to produce desirable encodings. In addition to the number of layers and nodes in a network, various constraints can be applied to enforce properties for the encoding [LZW18]. There are many ways of enforcing sparsity on the encoding [LZW18]. In this thesis, l_1 regularization is used to enforce sparsity on the encodings [JZZX13]. The l_1 regularization is defined in Definition 9.

Definition 9. Let α be a scalar called regularization factor, n the number of nodes on a layer of neural network. Now the l_1 regularization can be defined as follows:

$$l_1 = \alpha \cdot \sum_{i=0}^{n} |x_i|$$

5 Anomaly Detection using an Autoencoder and Density Based Outlier Scores

In this chapter, a method for calculating an outlier score for data points in a high dimensional data set using autoencoders is proposed. The chapter first provides an overview of anomalies and anomaly detection and then moves to describe the parts of the proposed method individually.

5.1 Anomaly Detection

An anomaly is a data object which significantly differs from the majority of data objects [Pan14]. The term anomaly has many synonyms like an outlier and a deviation which are used in various contexts [Pan14]. In this thesis, the terms outlier and anomaly are used extensively. In anomaly detection, the goal is to identify these distinct data objects from the rest typically without any supervision. Anomaly is not a binary property of a data point but is best modelled as a score which indicates how isolated the data point is from the rest [BKNS99]. A metric which is used to measure the anomalousness of a data point is called an outlier score. There are multiple techniques for anomaly detection, for example, model-, proximity- and density-based techniques. In this thesis, a novel method which combines both model and density-based techniques is presented. [Pan14]

In a model-based approach, a data object is identified as an anomaly if it does not fit the model well. The model-based approach requires the model to describe the normal cases accurately so that anomalies can be separated from the normal cases. One practical tool for model-based approach is naturally neural networks. Neural networks have been successfully applied for anomaly detection in domains such as aviation [NS16], computer systems [BCMLK16] and maritime [NVH$^+$18]. In case of a neural network model, the data points which are far from their prediction are considered as anomalies.

5.2 Autoencoding as a Model-Based Technique

Autoencoder neural network architecture presented in Chapter 4 is well suited for model-based anomaly detection in an unsupervised problem setting and autoencoders have been used for this purpose in [ZP17] and [AC15]. The decoder of the autoencoder creates a reconstruction of the original data. The difference between the original data and its reconstruction can then be used as an outlier score since the reconstruction error describes how well the autoencoder is able to capture the essence in the data point and how well the data point fits the autoencoder model. The outlier score defined by the absolute difference between the data point and its reconstruction is referred to as reconstruction error. The reconstruction error has been formalized in Definition 10. [Agg17]

Definition 10. Let D be a data set, and D' be a data set reconstructed from D via an autoencoder. Then the reconstruction r error is defined to be the absolute difference between the original data point $X_0 \in D$ and its reconstruction by the autoencoder $X_1 \in D'$.

$$r = |X_0 - X_1|$$

5.3 Density Based Technique for Analysing of Encodings

Using the reconstruction as the sole metric for anomaly detection assumes that the autoencoder is not able to capture the structure of most of the anomalies in the data. However, depending on the data set, this might not be sufficiently feasible. Autoencoders have the ability to model highly non-linear structure in the data, and at least for some anomalies, it might be able to create accurate reconstructions and therefore yield low reconstruction error also for some anomalies. The ability of autoencoders to fit the model also to anomalies has been used as an advantage in multiple research in the form of a dimensionality reduction technique which can also capture anomalies efficiently. When the autoencoder is able to capture the structure of the anomalies as well as the non-anomalous data points, the encodings produced by the autoencoder can be used to detect anomalies using density-based methods. This approach has been successfully used to detect anomalies in [PVD$^+$17].

In density-based techniques, outlier score is based on the density of objects on the area of the data point being considered. Sophisticated density-based techniques also take into consideration the density of the neighbouring data points when calculating an outlier score. Density-based techniques require a distance metric which is used to calculate a distance between any two data points. The use of a distance metric makes density-based techniques prone to the curse of dimensionality. [Pan14] [BKNS99]

The curse of dimensionality refers to problems which arise when dealing with high dimensional data. An example is increased time complexity which is the context in which the curse of dimensionality is described in [Cla94]. In the context of clustering and calculating outlier scores, the problem which arises is the average increased distance between any two data points. Distance between data points is central for calculating outlier scores, and increased distance between data points can change the situation so that points do not have any near neighbours. Increased distances can also make the distances between data points increasingly homogeneous and therefore make it more difficult to detect anomalies.

Due to the curse of dimensionality, it is beneficial to try to keep the sizes of the encodings produced by an autoencoder as small as possible while not sacrificing too much expression power of the encoding.

5.4 OPTICS-OF Density Based Technique

In this thesis, OPTICS-OF (*Ordering Points to Identify the Clustering Structure with Outlier Factors*) method [BKNS99] is used for calculating outlier scores for encodings constructed by an autoencoder. It is based on a clustering technique OPTICS (*Ordering Points to Identify the Clustering Structure*) [ABKS99] [BKNS99].

The OPTICS-OF method takes into consideration the local cluster structure of the data set. The sense of locality allows the method to work effectively on data sets with clusters of varying density when compared to other density-based approaches which use global parameters for cluster density such as DBSCAN. The concept of local

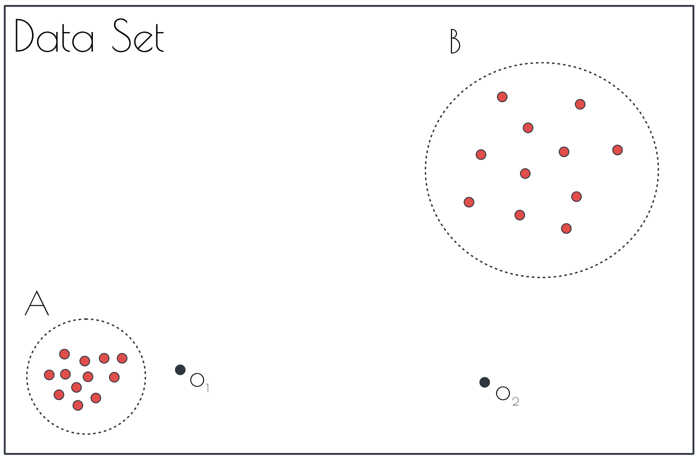

Figure 11: The figure illustrates a data set with a varying density. Red dots present normal data points and dark dots o_1 and o_2 present anomalies. There are two clusters within the data set A and B.

density is visualized in Figure 11. The cluster A has considerably larger density than cluster B. The data point o_1 is an anomaly since it has a considerably lower density than its local neighbours. However, if a global parameter describing the density of the whole data set would be required. It would then become very difficult to detect anomalies such as o_1. The data point o_1 has neighbours within roughly the same distance as data points in cluster B and therefore has roughly the same density as data points in cluster B. So without taking into consideration the locality of the data point it is hard to detect anomalies other than the ones which are far from all other data points such as o_2. Next, the details of OPTICS-OF method are presented. The presented definitions are from the paper [BKNS99] (markings vary partly from the original).

The Definition 11 gives a definition for neighbourhood of data point p which contains all data points within a distance at most ϵ from p. A distance from one of k-nearest neighbours is formulated in the Definition 12. Using Definitions 11 and 12 a definition for k-nearest neighbourhood of a data point p is formulated in Definition 13. [BKNS99]

Definition 11. Let ϵ be a positive real number, D a data set and d a distance measure in the data set D. The $\epsilon -$ neighborhood for $p \in D$ is defined as

$$N_{\epsilon\text{-neighbourhood}}(p) = \{o \in D \mid d(p,o) \leq \epsilon\}$$

Definition 12. Let k be a natural number and D a data set. Then for a data point $p \in D$ k-distance can be defined as
k-distance$(p) = d(p,o)$, where $o \in D$ is such that at least for k objects $o' \in D$ it holds $d(p,o') \leq d(p,o)$ and for at most $k-1$ objects $o' \in D$ it holds that $d(p,o') < d(p,o)$.

Definition 13. Let k be a natural number and D a data set. Then k-neighborhood for data point $p \in D$ can be defined as

$$N_k(p) = \{o \in D \mid d(p,o) \leq \text{k-distance(p)}\}$$

The k-nearest neighbourhood contains all data points which are within a distance of k-nearest neighbor from the data point p (see Figure 12). The k-nearest neighbourhood of a data point can contain more than k data points in case it has more than one neighbours exactly at *k-distance*. An example of this is visualized in Figure 12 where data points o_1 and o_2 are exactly *k-distance* from data point p.

The k-nearest neighbourhood is used in outlier score calculations as a local neighbourhood of a data point, and when determining the outlier score of a data point, its is density compared against the densities of data points within its k-nearest neighbourhood. The value of k is the sole parameter of the method, and it is marked with *MinPts* which is a shorthand for Minimum Points and refers to the minimum number of points to form a neighbourhood. [BKNS99]

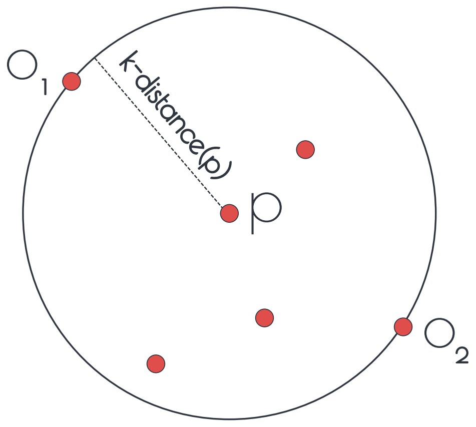

Figure 12: The k-nearest neighborhood of data point $p \in D$. In the illustration the size of k-nearest neighborhood is larger than k since both $o_1, o_2 \in D$ are exactly *k-distance(p)* from p.

In the OPTICS and OPTICS-OF methods the core and reachability distances are used as a metric for the local density of the cluster [ABKS99] [BKNS99]. The core distance of data point $p \in D$ is the smallest distance $\epsilon' < \epsilon$ which contains at least *MinPts* data points (see Definition 14). If such ϵ does not exist, then the core distance is not defined for data point p [BKNS99].

Definition 14. Let ϵ be a positive real number, *MinPts* a natural number and $p \in D$ a data point. Then the *core-distance* denoted as υ is defined as

$$\upsilon_{\epsilon,\text{MinPts}}(p) = \begin{cases} \text{Undefined}, & |N_\epsilon(p)| < \text{MinPts} \\ \text{MinPts-distance}(p), & \text{otherwise} \end{cases}$$

The reachability distance between two objects in a data defines the smallest distance, which makes two data points directly density-reachable [BKNS99]. Meaning the smallest distance required to make a point p part of the neighbourhood of point o. Since to create a neighbourhood at least *MinPts* points are required, the reachability distance can not be smaller than the core distance of the core point in the query. The reachability distance is defined in Definition 15.

Definition 15. Let ϵ be a real number, *MinPts* a natural number and $p, o \in D$ data points in data set D. Then *reachability-distance* denoted as ψ is defined as

$$\psi_{\epsilon,\mathrm{MinPts}}(p,o) = \begin{cases} \text{Undefined}, & |N_\epsilon(o)| < \text{MinPts} \\ max(\upsilon_{\epsilon,\mathrm{MinPts}}(o), d(p,o)), & \text{otherwise} \end{cases}$$

The local reachability distance describes the reachability distance of a point p with respect to its neighbourhood. Using local reachability distance a calculation for outlier factors (later in this thesis referred to as outlier scores) can be defined by comparing local reachability distance of a point to the points in its neighbourhood. The local reachability distance is defined in Definition 16 and outlier factor calculation in Definition 17.

Definition 16. Let $p \in D$ be a data point in data set D and $MinPts$ a natural number. Then the *lrd* (*local reachability density*) of a data point p is defined as

$$\mathrm{lrd}_{\mathrm{MinPts}}(p) = 1/\frac{\sum_{o \in N_{MinPts}(p)} \psi_{\infty,\mathrm{MinPts}}(p,o)}{|N_{\mathrm{MinPts}(p)|}}$$

Definition 17. Let $p \in D$ be a data point in a data set D and $MinPts$ a natural number. Then the OF_{MinPts} *outlier factor* (*outlier score*) is defined as

$$\mathrm{OF}_{\mathrm{MinPts}}(p) = \frac{\sum_{o \in N_{MinPts}(p)} \frac{\mathrm{lrd}_{\mathrm{MinPts}}(o)}{\mathrm{lrd}_{\mathrm{MinPts}}(p)}}{|N_{MinPts}(p)|}$$

5.5 Combining Outlier Scores

The reconstruction error of an autoencoder can be treated as an outlier score, and the OPTICS-OF method can be used to calculate an outlier score for the encodings produced by the autoencoder. Outlier scores from separate methods can be combined to form an overall outlier score for a data point [AS15]. Next, a general procedure designed for the combination of outlier scores is presented. In this thesis it is used to combine the reconstruction error and OPTICS-OF outlier scores.

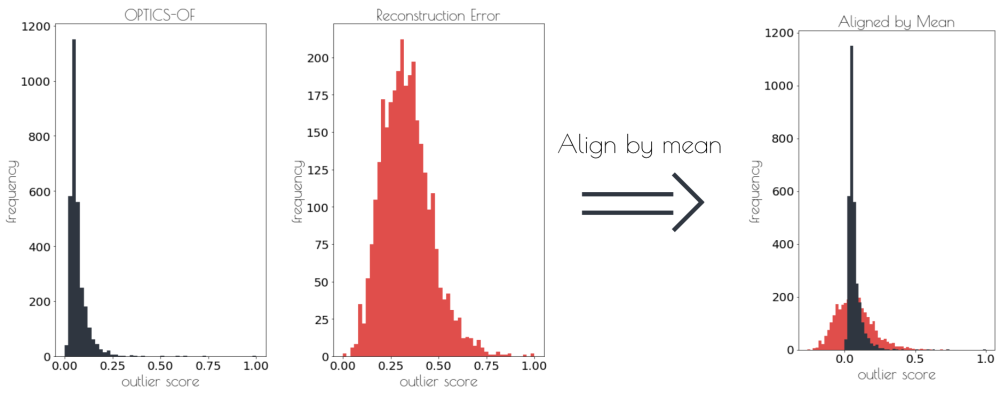

Figure 13: The figure presented the procedure of aligning normalized outlier scores by mean before combination. The histograms on the left present OPTICS-OF outlier scores of the encodings and the reconstruction errors. The rightmost histogram visualized both outlier scores after they have been aligned by mean.

The outlier scores produced by the different methods need to be normalized before combination [AS15]. Otherwise, different scales in outlier scores would cause some of the outlier scores to dominate the final combined outlier score over others. In addition to normalizing the outlier scores, the scores are shifted so that they share the same average. The operation is illustrated for reconstruction error and OPTICS-OF outlier score in Figure 13. The intuition for the shift operation is that depending on the data set and other parameters, the outlier scores have different abilities of detecting anomalies and the distribution of outlier scores differs. However, since the vast majority of the data points in the data set are non-anomalous points, the average of the outlier score reflects the outlier score received by non-anomalous points. This can be used to align the outlier scores properly.

Two standard formulas for combining outlier scores are average and max functions [AS15]. For combining the reconstruction error and OPTICS-OF outlier score, the max function has better properties than the average function, and therefore in this thesis, max-function is used for combining the reconstruction error and OPTICS-OF outlier score. The max is considered better since it is enough that either reconstruction error or OPTICS-OF outlier score is large for a data point for it to be considered an anomaly. There is a dependency between the reconstruction error and the OPTICS-OF outlier score since OPTICS-OF outlier score is calculated from the encodings. The reconstruction error describes the accuracy of the reconstruction and therefore reflects the quality of the encoding for a data point. When the reconstruction error is large, it is not relevant if the OPTICS-OF outlier score is small since the encoding does not describe the data point well. Vice versa, if the OPTICS-OF outlier score is large and reconstruction error is small the overall outlier score should be large since the encoding indicates that the data point is an anomaly. Because of this dependency, the average function is not deemed desirable for combining reconstruction error and OPTICS-OF outlier scores.

To optimize the performance of the reconstruction error - OPTICS-OF combined outlier score, a more sophisticated function might be required. However devising such a function requires insight into the behaviour of reconstruction error, OPTICS-OF and the autoencoders behaviour with data sets containing anomalies. In this thesis, the aim is to construct this insight and the development of more sophisticated combining function is left for future research.

6 Overview of Data and Feature Selection

In this thesis, the method introduced in Chapter 5 is used to detect anomalies in two data sets. This chapter provides an overview of those data sets and the features which were derived from the data.

6.1 Musk Anomaly Detection Benchmark Data Set

In order to evaluate the efficiency of the method and its' components presented in Chapter 5, a benchmark data set which contains labels differentiating anomalous and non-anomalous data points are used. A modified Musk data set originally from UCI Machine Learning repository was chosen as a benchmark data set. The data set was acquired through the ODDS library [Ray16]. The original data set was modified for the research performed in [AS15] where it was also used for evaluating anomaly detection methods. The same modified data set is used in the research of this thesis.

The Musk data set was chosen from the available data sets since it has a relatively high number of features 166, and it has also been used in existing anomaly detection research [AS15]. The high number of features provides an opportunity to evaluate the method of Chapter 5 with a high dimensional data set. The fact that the data set has been used as a part of existing research from the anomaly detection domain provides validation of the quality of the data set for the purpose of evaluating the performance of the method.

The Musk data set contains 3062 data points and has 166 features. The data set contains 97 anomalies which are 3.2% percentage of the whole data set. Each data point has a label which indicates whether the data point is an anomaly or not but these are used only for evaluation. The non-anomalous data points of the data set are from non-musk classes j146, j147, and 252, while the anomalous data points are from musk classes 213 and 211. [AS15]

6.2 Maritime Anomaly Detection using AIS Data

The anomaly detection method of Chapter 5 is used to detect anomalies in real-life data set of ship trajectories. The AIS data (*Automatic Identification System*) consists of dynamic GPS data and rarely changing static metadata [PVD$^+$17]. The dynamic GPS data consists of a series of longitude and latitude coordinates coupled with a timestamp and a mmsi (*unique identification*) of a ship which the coordinate belongs. The metadata contains general information about each ship, for example, name, type and size. The AIS data is used by ship tracking services, and it is the primary method for collision avoidance in maritime traffic [PVD$^+$17]. According to the International Maritime Organization's International Convention for the Safety of Life at Sea, all passenger ships and ships with gross tonnage 300 or more are required to have an AIS transmitter on a ship [PVD$^+$17].

The research of this thesis uses AIS data which contains data for ships near Finland from 30.10.2019 until 14.01.2020. The data points were filtered to include only ships in the area of interest which was selected to be the Gulf of Bothnia. This research limits the investigation to cargo and tanker ships. The cargo and tanker ships are mostly larger commercial ships which follow shipping routes. Their potential deviation from normal behaviour is considered to be interesting. The Figure 14 shows heat map images of the data before any filtering, after filtering by position to include on the Gulf of Bothnia and after filtering by ship type. When comparing the position filtered image (centre image) to the image after ship type filtering (rightmost image), it is visible that many trajectories travelling along the coastline have been filtered out. The ship type filtered image also contains considerably fewer trajectories which move on areas with little traffic since the image shows slightly fewer trajectories with very dark red colouring indicating some but little amounts of traffic on the area.

Ships near and at ports behave differently than ships at open sea. At ports ships are often required to make sharp turns, move at a slow speed and are subject to towing. The movement patterns commonly occurring at ports are anomalous behaviour when occurring

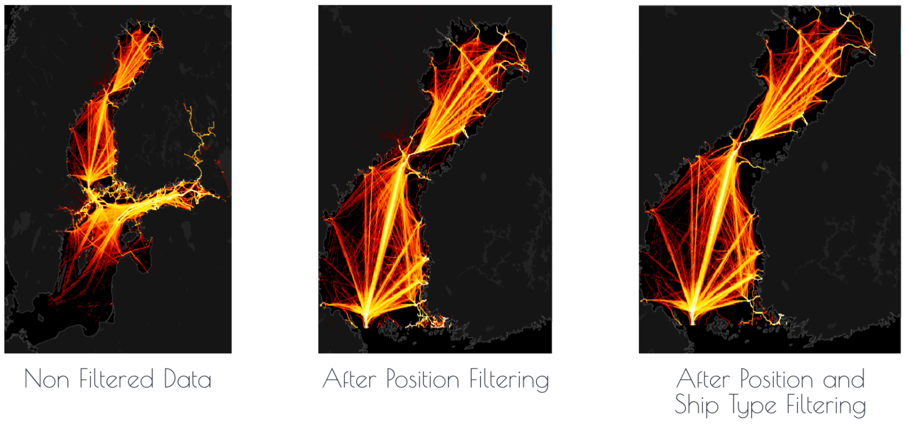

Figure 14: The leftmost image in the figure presents the whole data set available for the research. The centre image displays the data set after data was filtered to include only the points in the area of interest Gulf of Bothnia. The rightmost image is the fully filtered data which only includes cargo and tanker ships.

at open sea. The research of this thesis focuses on finding anomalies occurring at open sea and therefore also data points at and at the virginity of ports are filtered out.

The AIS data contains data points consisting of locations of each ship coupled with a timestamp. The data points are divided into trajectories in the same manner as in [YZZ+18] with slight variations. The following list contains the rules which were used to group AIS data points to trajectories:

- Two hours is the maximum time difference between any two consecutive AIS data points in the same trajectory. In case the time difference between two data points is larger the trajectory is divided into two separate trajectories.
- Max length for any trajectory is set to two days. If the trajectory is longer than two days, it is divided so that each is at max two days long.

After the AIS data points had been grouped into trajectories were filtered with the following conditions:

- A trajectory must contain at least five data points.

- A trajectory must contain a time span of at least an hour.

If a trajectory does not fulfil the specified conditions, it is considered not to have enough information to be useful.

The frequency of the data points in the original data set varied but for large proportions was around one sample per 5 minutes. The 5-minute frequency made the trajectories very long and caused many problems in the training phase, such as not being able to fit training data into GPU memory and very long training times. As a solution, the data points were downsampled to one data point every 15 minutes. Since cargo and tanker ships are large ships changes in their movement happens mostly slowly therefore 15 minutes was deemed acceptable time interval.

Two distinct feature sets are derived from the AIS data and used separately for anomaly detection. Research in [YZZ+18] introduces location-agnostics features. In [YZZ+18] encodings produced by an autoencoder from the location-agnostic features are successfully used to cluster ship trajectories showing that the derived features are meaningful. Location-agnostic features capture the shape of a trajectory and movement of a ship along it. The features do not take a stand on the location of the trajectory [YZZ+18]. The location agnostics features use a sliding window to extract a behaviour sequence from each of the trajectories. Set of trajectories from which behaviour sequences were generated was expanded from the original set of trajectories by including their subsets of 4, 12 and 24 hours. Therefore some of the behaviour sequences are overlapping. For each step of the sliding window, a behaviour sequence entry is formed by first calculating average speed, acceleration and change of rotation [YZZ+18]. Then the final behaviour sequence entry is formed by calculating the following for each of the properties: $\{mean, max, 75\%\text{-}quantile, 50\%\text{-}quantile, 25\%\text{-}quantile, min\}$. The values are calculated directly using longitude and latitude, and also an approximation is made that longitude and latitude are located on a flat surface. This research uses the same features with the same approximation since [YZZ+18] showed them to be successful as a feature set for analyzing ship trajectories.

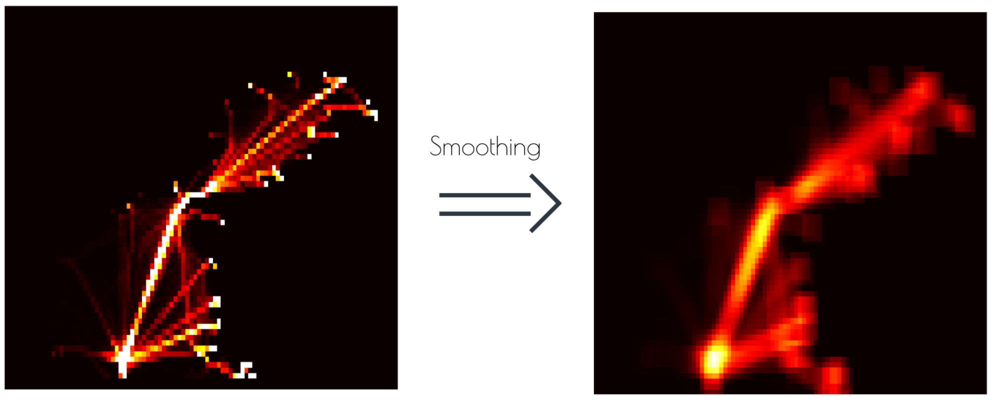

Figure 15: On the left, in the figure, there is a rasterized version of a heat map before smoothing. The smoothed version of the rasterized heat map is presented on the right side of the figure.

The other feature set is not location-agnostic but instead uses the location of trajectory heavily. The heat map captures the locations of the common shipping lanes. Since shipping lanes are considered an essential feature for the movement of cargo and tanker ships, the location of each point on a trajectory was given a heat score based on its position on the heat map. In order to not include minor details on the heat map, the map was rasterized into a grid where the size of each cell is 0.1 times 0.1 longitude and latitude angles. The rasterized heat map has cells containing vastly more points than some of the other cells. In order to take into account that a shipping lane is considered normal as long as it has a significant amount of traffic, a reasonable max value was set to cells with huge heat scores. In the rasterized heat map, there are lots of areas where the heat scores between neighbouring cells vary greatly. Feature values which exhibit sudden dramatic changes proved to be problematic in the training phase of the research. In order to moderate the dramatic changes, a smoothing procedure was performed on the heat map. In the smoothing phase, value for each cell was selected to be the mean of it and the cells within two Manhattan distance from it. The Figure 15 shows the rasterized heat map before and after the smoothing operation. The other features for the location wise feature set includes the actual longitude and latitude, heading and the previously described heat score. The location wise sequences were chosen to present six hours long time intervals where each sequence consists of 24 data points with intervals of 15 minutes. In order to hide the cuts in trajectories of six-hour intervals, sequences were generated using a sliding window which moved was always moved only half of its length. The sequences of the feature set are therefore partially overlapping.

6.3 Cargo Ship Engine Failure as Known Anomaly

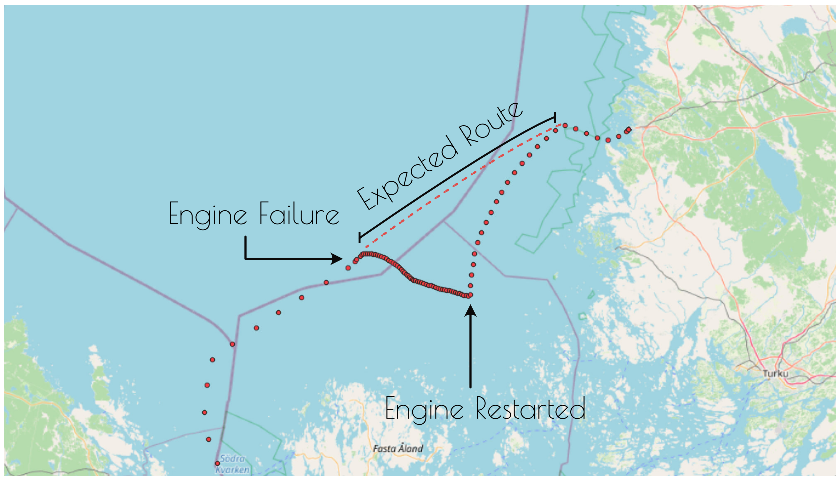

Figure 16: The anomalius trajectory where the ship Sampogracht experienced an engine failure. The engine failure and restart have been marked to the image as well as the expected route (dashed line).

The data set contains one initially known anomaly. The known anomaly servers both as an example of an anomaly in maritime traffic which the research aims to locate and as a case for which more detailed analysis can be performed. A Dutch cargo ship Sampograft (see Figure 17) drifted on the sea after experiencing engine failure on the night between 12.01.2020 and 13.01.2020 [ML20]. The trajectory of Sampograft during the 12th and 13th of January is visualized in Figure 16. It is visible that

Figure 17: Cargo Ship Sampogracht experienced an engine failure during 12th and 13th of January 2020 and drifted on the Gulf of Bothnia.

the engine failure caused the ship's trajectory to deviate from typical trajectories in the area. The expected trajectory is marked in the images with a dashed line. The engine failure also caused the ship to slow down significantly, which can be seen as the distance between measurements getting smaller during the time the engine was not used. Also, sudden drifting behaviour could be considered abnormal in general.

7 Methodology

This chapter presents the research design and used neural network architectures and implementation details. Research design includes the data normalization and data split strategy. The chosen neural network architectures are presented with justifications.

7.1 Research design

The goal of this research is to evaluate the anomaly detection method presented in Section 5 and identify the properties of its components. The proposed method is an extension of the method presented in [PVD$^+$17]. In the method, an autoencoder is used to detect anomalies by combining reconstruction error and OPTICS-OF outlier score calculated from the encodings. The proposed method is evaluated using the Musk data set described in Chapter 6.1. The method is then applied to two feature sets derived from a real-life AIS data set described in Section 6.2. Both the Musk and AIS data set are treated in an unsupervised manner. The labelling in the Musk data set is used solely to evaluate the performance of the method and is not seen by the models. The AIS data set contains one trajectory known to be anomalous described in Section 6.3, which is used to evaluate and analyze the behaviour of the proposed method.

Autoencoders have been used to detect anomalies in existing research with successful results in [PVD$^+$17]. Also, encodings produced by an autoencoder were used in [YZZ$^+$18] to cluster trajectories. The research in [YZZ$^+$18] provides proof that an autoencoder can be used to construct meaningful encodings from features derived from AIS data. In [PVD$^+$17] AIS was used in combination with over-the-horizon radar to detect anomalies from encodings. In [PVD$^+$17] anomalies are identified from the encodings using a density-based technique OPTICS. The research of this thesis takes a similar approach to analyze encodings by using a density-based technique OPTICS-OF which is based on OPTICS for identifying anomalies from the encodings.

Reconstruction error of an autoencoder is identified as an important metric for detecting anomalies in [Agg17]. Since outliers are potentially more resistant to compression than non-anomalous data points [Agg17] reconstruction error can be used to detect potential anomalies. The reconstruction error indicates the quality of any particular encoding, and therefore, there exists a dependency between the reconstruction error and the OPTICS-OF outlier score. The max function is used to combine reconstruction error and OPTICS-OF outlier score to produce a single outlier score as described in Section 5.5.

The Musk data set contains labelling, which indicates which data points are anomalies. The labelling allows the construction of a training and test sets which contain various proportions of the anomalies. Autoencoder models are trained with the constructed data sets and performances of reconstruction error, OPTICS-OF and the combined outlier score are calculated. Evaluating the models' performances when they have been trained with seeing different proportions of the anomalies in the training phase can be used to evaluate the performance of the method on unseen anomalies and robustness not to overfit on individual anomalies in the training data.

The proposed model is then applied to the two feature sets of the AIS data set to see whether it is able to identify the known anomaly described in Section 6.3 and to find yet unknown anomalies. The AIS data set is a more complex one than the Musk with sequence to sequence prediction problem setting and high dimensional features.

7.2 Normalization, Train - Test Data Split, Loss Function and Hyperparameters

The data points were normalized to aid and speed up the learning process on all feature sets. The Musk data was normalized using standard min-max scaling. On the AIS data set, both location-wise and location-agnostic features were normalized. A min-max scaling was used for the speed features. Longitude and latitude positions were scaled with min-max scaling where min and max were taken from the enclosing bounding box of the area of interest.

The feature sets were divided into three parts train, test and validation sets. On the Musk data set division was done separately for anomalies and non-anomalous data points in a randomized fashion. Then the final data sets containing specified proportions of anomalies in the training set were formed. On the AIS data, some of the sequences are overlapping; for this reason, it was not possible to make division by selecting sequences at random. If sequences would have been selected randomly overlapping sequences might have ended up in different sets resulting into a situation where a data point might have been presented both in a training and test set. Therefore the split on the AIS data was done by diving the trajectories and corresponding sequences by date. The test set was chosen to be from 1sth to 10th of December 2019, validation set from 11th to 15th of December 2019 and the training data the rest of the data.

The loss function is chosen to be MSE (*Mean Squared Error*) defined in Definition 2. In the training of an autoencoder, the goal of the training is to minimize the reconstruction error. MSE can be used to minimize the Euclidean distance between the reconstruction and the original input and is therefore considered to be a good fit for the purpose. Even though in the training of the model the goal is to minimize the MSE and reconstruction error this is not the overall goal the model is used for. In Figure 18 it can be seen that the precision of the model for anomaly detection and the MSE do not correlate. In order for the model to be useful it must be able to capture some relevant patterns in the data but as the figure shows lower MSE does not guarantee better anomaly detection results. Since anomalies in the data set are not known it is not possible to use anomaly detection precision as a loss function.

The primary hyperparameters of the research are batch size and learning rate. Multiple learning rates and batch size were tested. Here the performance means performance on anomaly detection. It is possible that some other hyper parameter values would have yielded better results but which were not found in the experiments. However it is difficult to measure the effect a hyperparameter has on an unlabeled data set. On batch sizes, the best performances were obtained using small batch sizes between one and ten. On the Musk data set batch size, the final models were trained with a batch size of five. On the feature sets derived from the AIS data the batch size was set to one. The learning rate was set to $1e{-}4$ for the models trained on feature sets of AIS data and to 1e−3 for the models trained on the Musk data.

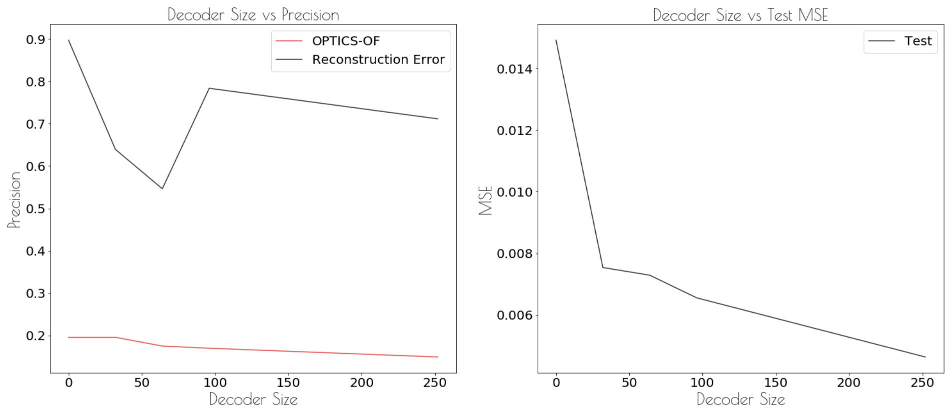

Figure 18: Precisions and Test MSEs plotted as a function of Decoder Size. The x-axis has the number of nodes in the decoder in addition to the output nodes, and the y-axis has the precision of OPTICS-OF and reconstruction error anomaly detection methods on labelled Musk data set.

7.3 Autoencoder with a Deep Encoder and a Shallow Decoder

In this thesis, autoencoders are used for dimensionality reduction and feature learning in an anomaly detection context. The goal is that the autoencoder learns high-level features of the data, which is then presented in the encoding analyzed using OPTICS-OF. Multiple autoencoder architectures with varying sizes and symmetries were tested on the Musk data set. The most successful architecture employed a deep encoder with multiple hidden layers and a shallow decoder with a single layer. The architectures were evaluated both by the performance of the OPTICS-OF and reconstruction error methods in detecting anomalies from the encodings and the level of reconstruction error on the Musk data set. Reasoning providing a possible explanation why the deep encoder with a shallow decoder proved to be successful is presented next.

In order to enable the encoder to learn high-level features, the encoder has multiple hidden layers. Since the encoding should hold relevant information of the original data, the process of constructing the original input from the encoding should not require multiple hidden layers. Not having multiple layers on the decoder reduces the complex logic the model stores via the weights in the decoder. Figure 18 shows how anomaly detection precision drops as more nodes are added to the decoder. There is a similar but weaker effect when nodes are dropped from the encoder. This might indicate that in general simpler models are more effective at anomaly detection. However more research would be required to verify this.

The AIS data is more complex than the Musk data. Therefore in its case, the decoder also has one hidden LSTM layer in addition to time distributed layer (on info about time distributed layer see Section 7.4).

The reasoning of this section was used to select the neural network architecture for the research. It is possible that some other architectures would have performed better. As with the hyperparameters, it is not trivial to optimize the architecture for anomaly detection performance since it can not be measured on an unlabeled data set. Therefore the same reasoning which was used on the Musk data set was also applied on the AIS data set.

7.4 Machine Learning Model Implementations

The machine learning models were implemented using the Tensorflow 2.0 library [AAB$^+$15] and trained on a GeForce GTX 1080 NVIDIA GPU. The data preprocessing used lots of other open-source libraries which at the time of writing are commonly used.

The Musk data set has features which are suitable for a feed-forward neural network. Autoencoders with Feed-Forward encoders and decoders were used in the research performed on the Musk data set. Multiple tests were performed to determine good architecture. The final architecture is presented in Figure 19. The design follows the reasoning presented in Section 7.3.

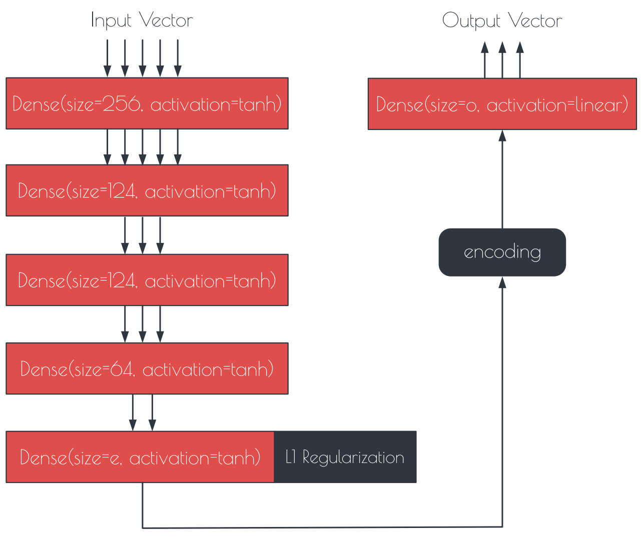
Input Vector
Output Vector
Dense(size=256, activation=tanh)
Dense(size=o, activation=linear)
Dense(size=124, activation=tanh)
encoding
Dense(size=124, activation=tanh)
Dense(size=64, activation=tanh)
Dense(size=e, activation=tanh)
L1 Regularization

Figure 19: The architecture of autoencoders used with the Musk data set.

On the feature sets of the AIS data, a sequence to sequence multilayer LSTM autoencoders are used to construct encodings of ship trajectories. The reason for using LSTM over, for example, a sliding window and a feed-forward neural network is the ability of LSTMs to observe temporal dependencies as described in Chapter 3. An LSTM was chosen over other recurrent neural network variations such as gated recurrent unit since in [YZZ+18] LSTM was observed to outperform gated recurrent unit when working with the location-agnostic feature set and LSTMs have better ability to detect long term dependencies than vanilla recurrent neural networks [YZZ+18]. Multiple different architectures were tested, and the final chosen architecture is shown in Figure 20. The architecture features the deep encoder and shallow decoder for which the reasoning was presented in Section 7.3.

The last layer on the AIS data autoencoders is time distributed dense layer. Time distributed layer runs the input through the same weights and layer for each time step. The result is a time series where each time step is an output vector of the time distributed layer.

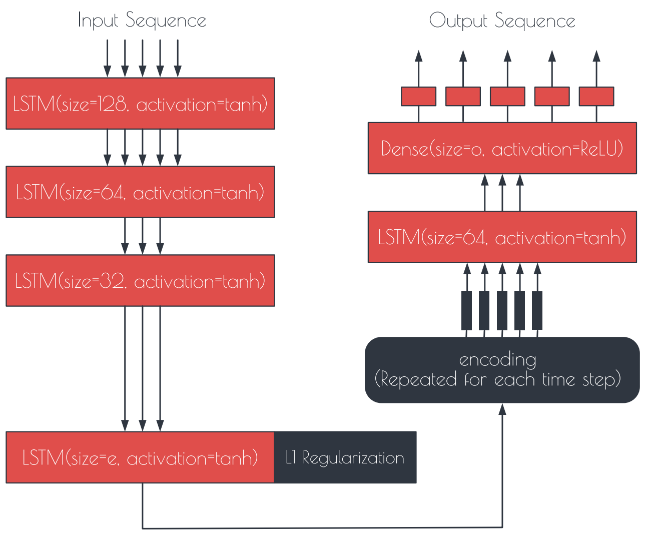
Input Sequence
Output Sequence
LSTM(size=128, activation=tanh)
LSTM(size=64, activation=tanh)
LSTM(size=32, activation=tanh)
LSTM(size=e, activation=tanh)
L1 Regularization
Dense(size=o, activation=ReLU)
LSTM(size=64, activation=tanh)
encoding
(Repeated for each time step)

Figure 20: The architecture of autoencoders used with the AIS data set.

8 Results and Discussion

In this Chapter, the main empirical results of the research are presented with accompanying analysis. The first section goes through the results on the Musk data set presented in Section 6.1. On the Musk data set, the anomalies are known, and this makes it possible to analyze how the amount of anomalies in the training data affects the model's ability to detect anomalies. The other focus area on the Musk data set results is to evaluate the performance of the reconstruction error, OPTICS-OF encoding analysis and the combined outlier score. We also aim to identify the characteristics of their behaviour.

The second section presents the results on the AIS data set presented in Section 6.2. Two distinct feature sets have been derived from AIS data, and the results are presented for both data sets. Here the focus is to present the results on the known anomaly presented in Section 6.3 for reconstruction error, OPTICS-OF encoding analysis and the combined outlier score. Lastly, some examples of the trajectories with highest outlier scores are presented coupled with general notions of which kind of trajectories received high outlier scores and which properties they had in common.

8.1 Musk Data Set

This section presents the results of the experiments performed on the Musk data set. First, the metric used for the evaluation is presented, followed by details of the training processes of the models. Then results of the anomaly detection are presented for each of the three methods: reconstruction error, OPTICS-OF and combination of the two scores. The numerical results are listed in Appendix A.

On the Musk data set 3.2% the data points with the largest outlier scores are selected as anomalies. This means that in the experiments performed on the Musk data the number of selected anomalies always equals to the true number of anomalies in the data. Now the performance of the models can be evaluated using a metric called precision. Precision is the number of true positives divided by a sum of true positives and false positives (see Definition 18). In the case of the Musk data set and the described anomaly selection scheme this calculates the percentage of the anomalies in the data points which the model assigns the highest outlier scores.

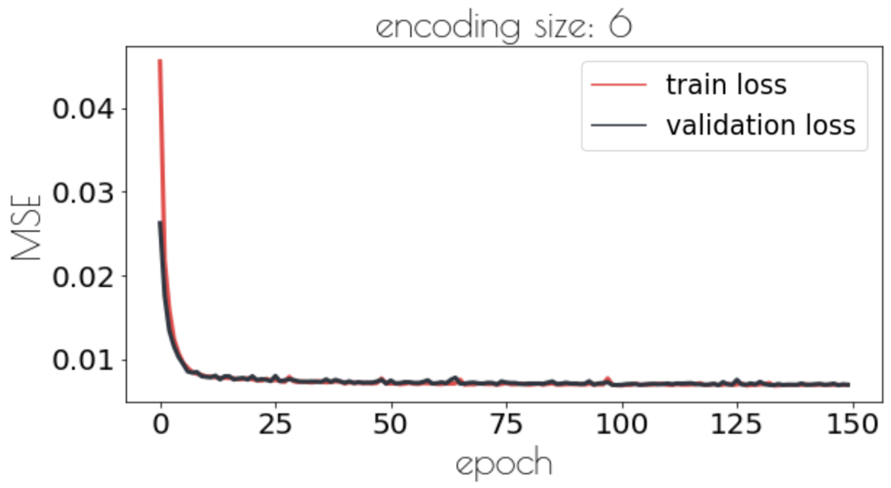

Figure 21: Visualization of training progress for the Musk data set. The model had an encoding size of 6 and was shown 40% of the anomalies during the training phase. The red line is the loss on the training set, and the grey line is a loss on the validation set.

Definition 18.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

For the Musk data set autoencoder networks with the same architectures were trained with various encoding sizes e, and the number of anomalies shown to the network was varied. The final MSE losses are listed in Table 2. The neural networks were trained for 150 epochs each. The training process proceeded in a similar fashion for all the trained models. The models learned most during the first few epochs after which the learning stalled. Validation data was also used to make sure that there was no significant overfitting. Since the loss is roughly the same on training, validation, and test sets, it can be concluded that the models have not suffered from overfitting. The training process for one of the models is visualized in Figure 21; the other models had very similarly shaped training curves.

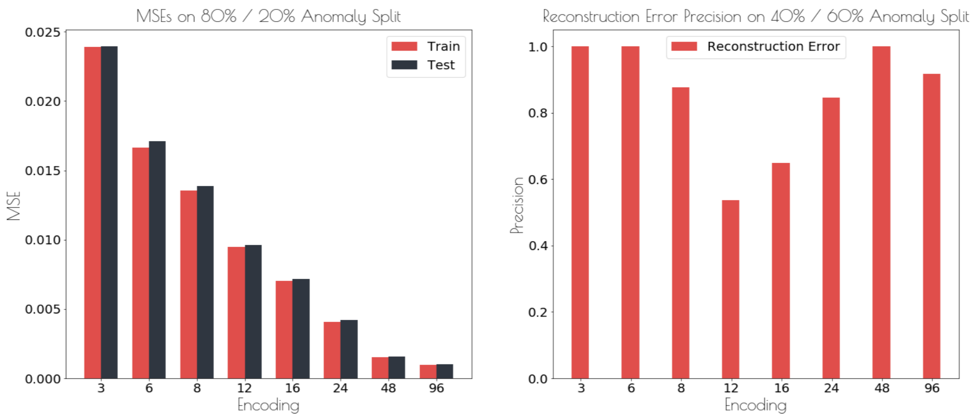

Figure 22: Correlations between encoding size and MSE and reconstruction error precision. The encoding size - MSE correlation is presented on the left chart and encoding size - reconstruction error precision on the right chart.

A clear correlation between the encoding size and the loss of the final model is evident in Table 2: as the encoding is increased the loss of the final model decreases. This is due to the amount of information the encoder is able to contain to the encoding, making it easier for the decoder to construct the original input. The described correlation is presented in Figure 22 on the (left).

Table 2 shows that varying the number of anomalies in the training data does not have a significant impact on the model's ability to capture the structure of the non-anomalous data points. This is pointed out by the fact that the MSE losses on the training set are very similar on all proportions of anomalies in the training set. However, the values indicate that when most of the anomalies reside in the test set the MSE loss on the test set is larger than on the training set. This confirms that the models can not capture the structure of anomalies using the same method as for non-anomalous data points since the trained model gives bad reconstructions for anomalies it has not seen during the training.

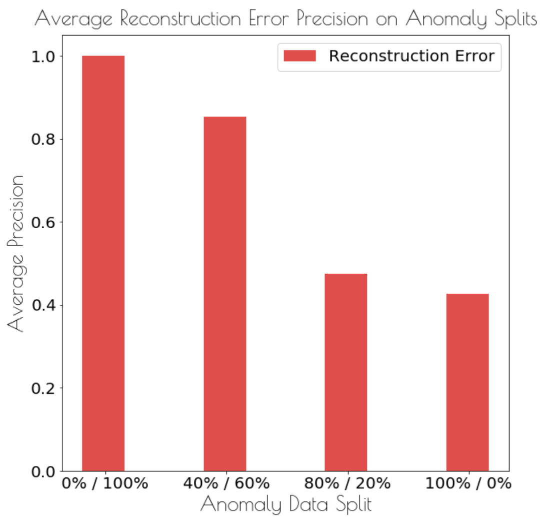

Figure 23: Correlation between the average precision of reconstruction error for the chosen encoding sizes and the proportion of anomalies in the training data.

The precisions of models for detecting anomalies using reconstruction error are displayed in Table 3. The table shows a clear trend that the amount of anomalies in the training data has a significant effect on the reconstruction error's performance on anomaly detection. When the models have not seen any anomalies, the reconstruction error is able to spot all of the anomalies in the Musk data set. This means that the models produce large errors for anomalies since they can not capture their structure, having only seen non-anomalous data points in the training phase. When more anomalies are shown to the models in the training phase, they learn to also reconstruct anomalies better and therefore, in overall the reconstruction error of the anomalies decreases. The Figure 23 presents the correlation between the proportion of anomalies in the training set and the reconstruction error's precision.

As described in Chapter 6, the anomalies in the Musk data set are taken from other classes of musk. Therefore some of the anomalies in the data set may be similar to each other since they could belong to the same class of musk even though they are different from the majority of data points. This might amplify the decrease in performance when larger proportions of anomalies are included in the training set since the model being trained can learn characteristics of the anomalies better if it has more than one similar case. Therefore it is not clear whether or not this effect would be present in case the anomalies stemmed from some more random data source, for example, measurement error.

Another correlation visible in Table 3 is the correlation between the encoding size and the precision of the model to capture anomalies. Small encoding sizes produce models which are good at detecting anomalies. When the encoding size is increased the precisions of the resulting models drop rapidly. However, when the encoding size is increased enough, the precision increases again. The efficiency of small encoding sizes in anomaly detection supports the claim that anomalies are more resistant to compression in the context of the model, which mostly sees non-anomalius data points. With small encoding sizes, the rate of compression is larger. It is not clear why the performance of models with large encoding sizes increases when encoding size is increased further. This phenomenon would require further research. The phenomenon is presented in Figure 22 (right).

The anomaly detection results for OPTICS-OF method are displayed in Table 4. The OPTICS-OF anomaly detection method was applied to the encodings produced by the models. The table shows the encoding sizes, *MinPts* parameters of the OPTICS-OF method and the number of anomalies shown to the model in the training phase. The numeric precision results indicate abysmal performance for the OPTICS-OF method on the Musk data set. However, looking closer which data points the OPTICS-OF flags as anomalies with different parameters reveals that with small values of *MinPts* OPTICS-OF produces large outlier scores to only a few data points from which a large proportion are actual anomalies. The described phenomenon is presented in Figure 24. Some values of *MinPts* have been omitted from the sequence but they closely follow the trend where OPTICS-OF starts to give increasing amounts of larger outlier scores when *MinPts* is increased and the amount of false positives increases rapidly. The same phenomenon can be observed regardless of the number of anomalies shown to the model in the training phase.

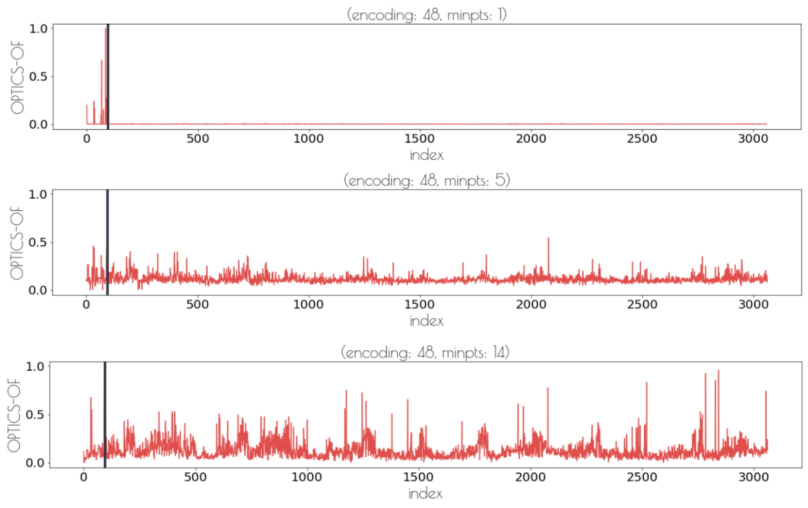

Figure 24: Outlier scores for each data point in the musk data set. The indices of data points are on the x-axis, and the outlier score given by OPTICS-OF is on the y-axis. The anomalies are located in indexes 0 - 96. A grey vertical line has been drawn to separate the anomalies and non-anomalius data points.

The OPTICS-OF outlier score and reconstruction error were combined, as described in Section 5.5. The results of the combined scores on Musk data set are presented in table 5. The table shows that when reconstruction error is performing poorly and the *MinPts* parameter has been chosen correctly combining the OPTICS-OF score with reconstruction error can result into a small increase in performance on the Musk data set in comparison to using just reconstruction error. The performance increase is due to OPTICS-OF being able to pick up a few anomalies which reconstruction error misses when performing poorly. The slight increase is most evident when 100% of the training samples were in the training set since that is when reconstruction error has the worst performance.

It is also noticeable that combining OPTICS-OF with reconstruction error on many occasions also lowered the performance when comparing to the performance of just reconstruction error. On the Musk data set reconstruction error performed extremely well and in many cases caught 100% of the anomalies and in these cases every miss classification from the side of OPTICS-OF potentially lowers the performance. When the value of *MinPts* was not optimal large drops in performance appear.

8.2 AIS Data Set

In this section results on the AIS, data set are presented. Two sets of features have been derived from the AIS data, as described in Chapter 6. The AIS data set has one known anomaly presented in Section 6.3, which is of particular interest. Since the known anomaly is only one case, it can not be used to evaluate the performance of the anomaly detection methods. This section focuses on presenting findings made using anomaly detection methods presented in Chapter 5 and aims to point out what characteristics sequences flagged as anomalies posses. The numeric results for location-wise feature set are presented in Appendix B and for location-agnostic features in Appendix C.

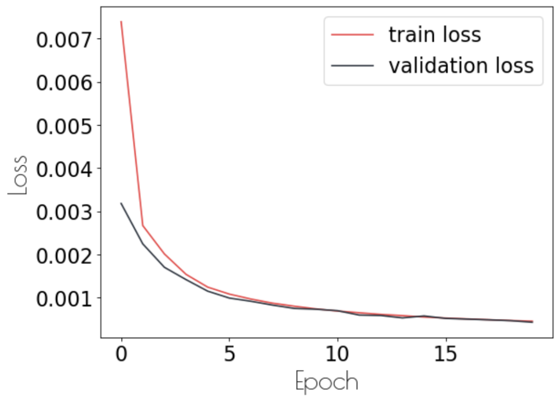

Figure 25: Training curve for a model trained with AIS location-wise feature set with an encoding size 8. The figure shows that most of the learning happened in the first few epochs.

Models were trained on the two feature sets with same architecture presented in Section 7.4. They were trained for 20 epochs each. For models trained on the location-wise data set the training to proceed by most of the learning happening on the first few epochs and then stalling. For the models trained on the location-agnostic data set the learning similarly stalled for some time after first few epochs but then suddenly experienced a massive increase in performance. The training

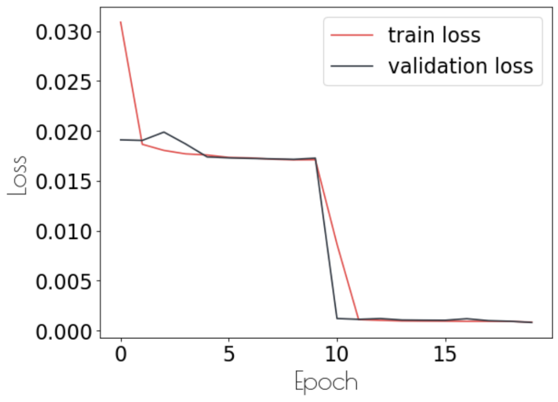

Figure 26: Training progress for a model trained with AIS location-agnostic feature set with an encoding size 8. The figure shows that there is a sudden increase in performance after the training had mostly stalled after the first few epochs.

process exhibiting the described behaviour is presented in Figure 26. It is not clear what causes this behaviour nor is it clear would a similar increase in performance also happen on the location-wise feature set after some amount of epochs or is this behaviour inherent for the location-agnostic feature set and the model type combination. It is worth noting that similar training behaviour was also present on models trained on location-agnostic features in [YZZ+18]. A training curve for one of the models trained on location-wise data set is presented in Figure 25.

On all the training curves on both data sets the validation loss and training loss are relatively close together and on the fully trained models losses on test and training sets are on the same level. This indicates that there has been no significant overfitting.

The table 6 presents MSE losses of the trained models on the location-wise feature set and corresponding results for models trained on location-agnostic feature set are in table 10. The training losses exhibit the same characteristics which were also evident on the models trained on the Musk data set: training losses decrease as the encoding size increases also as is common test set has slightly higher loss than the training set.

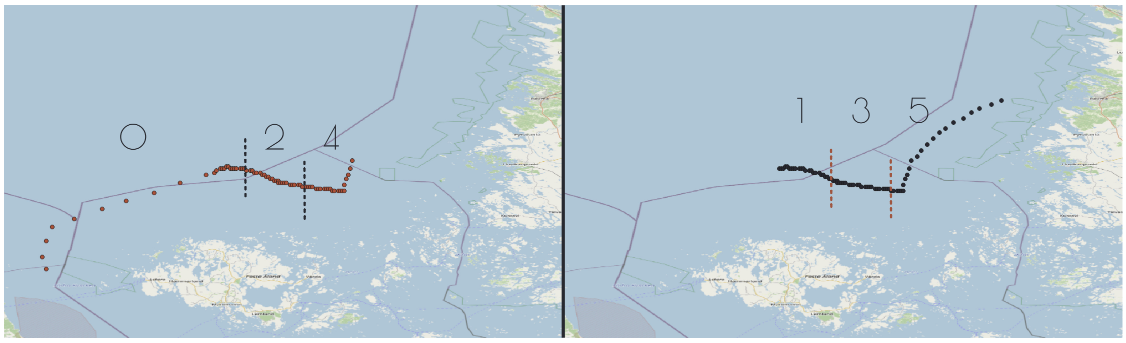

Figure 27: Sequences belonging to the known anomaly and indicates an index for the sequences used in the location-wise features results in tables.

The Figures 27 and 28 define indexes for the feature sequences of location-wise and location-agnostic feature sets which are part of the anomaly of interest. The reconstruction error, OPTICS-OF and combined outlier score and presented for each of these sequences. Some of the sequences are partially overlapping for the reasons described in Chapter 6.

Tables 7 and 11 present the results of anomaly detection using reconstruction error for the known anomaly. The tables list the percentage of sequences in the feature set, which have a larger outlier score than the feature corresponding to index. Meaning the smaller the value, the more sequence is considered to be an outlier. For example, on location-wise features on a model with encoding size eight 0.1% of the sequences have higher outlier scores than sequence at index zero in the anomaly of interest.

On location-wise features, the sequence at index zero has the largest reconstruction error of the sequences in the known anomaly. The sequence at index zero has a smooth turn which is normal ship movement in the area, and it contains the slow of ships speed as the engine failure occurred. Out of the sequences belonging into the known anomaly's trajectory the ones at index zero, and five get the largest scores. They both display smooth shape and large variations in speed which attracted high outlier scores from reconstruction error in general. Other sequences in the known anomaly contained drifting of the ship.

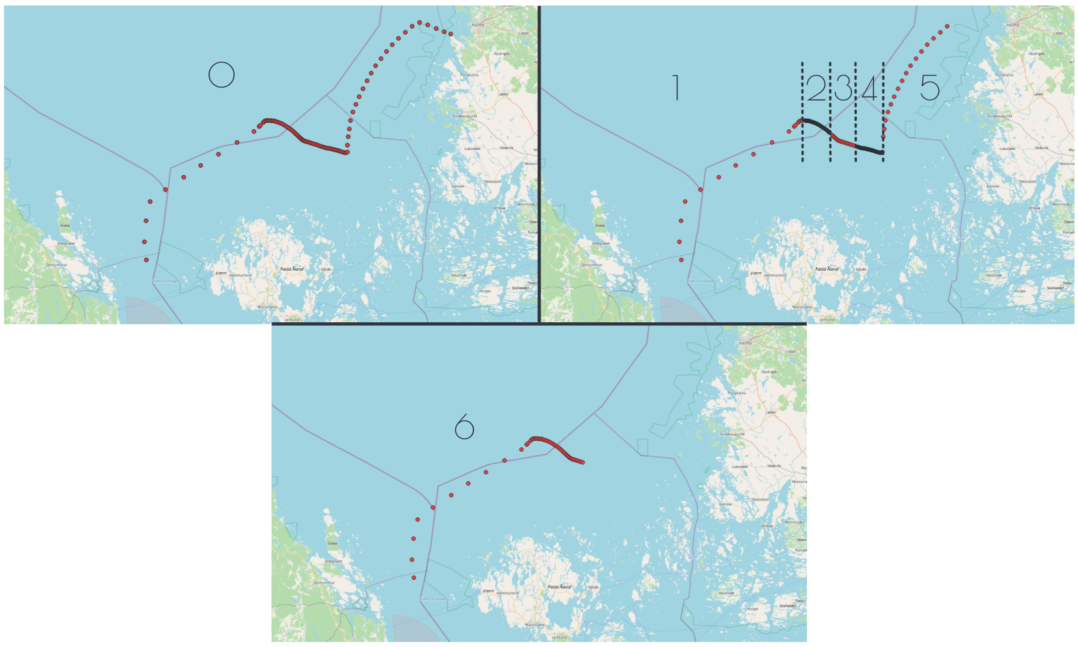

Figure 28: Sequences belonging to the known anomaly and indicates an index for the sequences used in the location-agnostic features results in tables.

These did not receive high outlier scores. The reason for this might be that the data set contained lots of similar slow speed trajectories; the main difference was that their location was usually nearer the coastline.

On location-agnostic features, the models assign large reconstruction errors to sequences at index zero and six. The reasoning for the characteristics which attract high scores from reconstruction error is the same as presented for reconstruction error on location-wise features.

The OPTICS-OF behaved similar way as with Musk data set in terms of a number of data points it assigned large outlier scores for. With small values of *MinPts*, few data points received large outlier scores and as *MinPts* was increased the number of data points with large outlier scores also increased. The phenomenon was inline with Figure 24 of the OPTICS-OF on Musk data set. The results of OPTICS-OF on the known anomaly on both location-wise and location-agnostic data sets varied greatly depending on the encoding size. The numeric results are presented in tables 8 and 12. On sequences of the known anomaly with location-wise features, OPTICS-OF is able to identify one or two sequences as potential anomalies on each encoding size, but it varies which ones. On location-agnostic features, the performance of OPTICS-OF is poorer.

The reconstruction error mostly dominates the combined outlier scores on the known anomaly on both feature sets. There are few exceptions on location-agnostic data set where combined outlier score performs significantly better than reconstruction error. Such cases are present for *MinPts* value 2 on *(encoding: 8, index: 3)* and *(encoding: 4, index: 1)*.

The Figure 29 shows some of the trajectories which the models trained on location-wise features assigned highest outlier scores. The reconstruction error was highest on trajectories with a large number of turns and smooth round shapes. The first image of reconstruction error anomalies shows a ship arriving at a port where ships navigate between islands. Many of the ships arriving at the particular port got high outlier scores because of the navigation manoeuvres which were required to avoid the islands. In general, these cases can be considered as miss-classifications. However, the chosen image shows also other unexpected behaviour since the ship has then quickly exited the port towards the north without entering the port. The image in the centre shows a trajectory where a ship has taken a circular route. The third image shows a trajectory where a ship has made two 180 degree turns on a short interval.

OPTICS-OF gave high outlier scores to trajectories which contained data points resulting from an anchored or otherwise stationary ship. Other qualities which attracted large outlier scores from OPTICS-OF were sharp turns and moving along the coastline. The first image from the right on OPTICS-OF row shows the trajectory of an anchored ship. Ships anchoring, for example, to wait for berths to become available at ports is common. Therefore these cases can be considered as miss-classifications. Next two images show examples of trajectories with multiple sharp turns which could be considered abnormal to a cargo ship.

The last row in the image presents findings from the combined scores. The combined score assigned highest values to some of the same trajectories as OPTICS-OF and reconstruction error. Anchored ships from OPTICS-OF also got high combined scores. Three images here show some of the finds from the combined score.

Figure 30 shows trajectories which received high outlier scores from models trained on location-agnostic features. The OPTICS-OF method produced poor results on location-agnostic features. The figure shows that OPTICS-OF assigned high outlier scores to long trajectories which moved along regular shipping lanes. A lot of the OPTICS-OF's poor performance also transferred to the combined score which therefore also had a poor performance. Reconstruction error performed better, finding some trajectories with odd shapes. On location-agnostic feature set reconstruction error miss-classified a large number of anchored ships as anomalies.

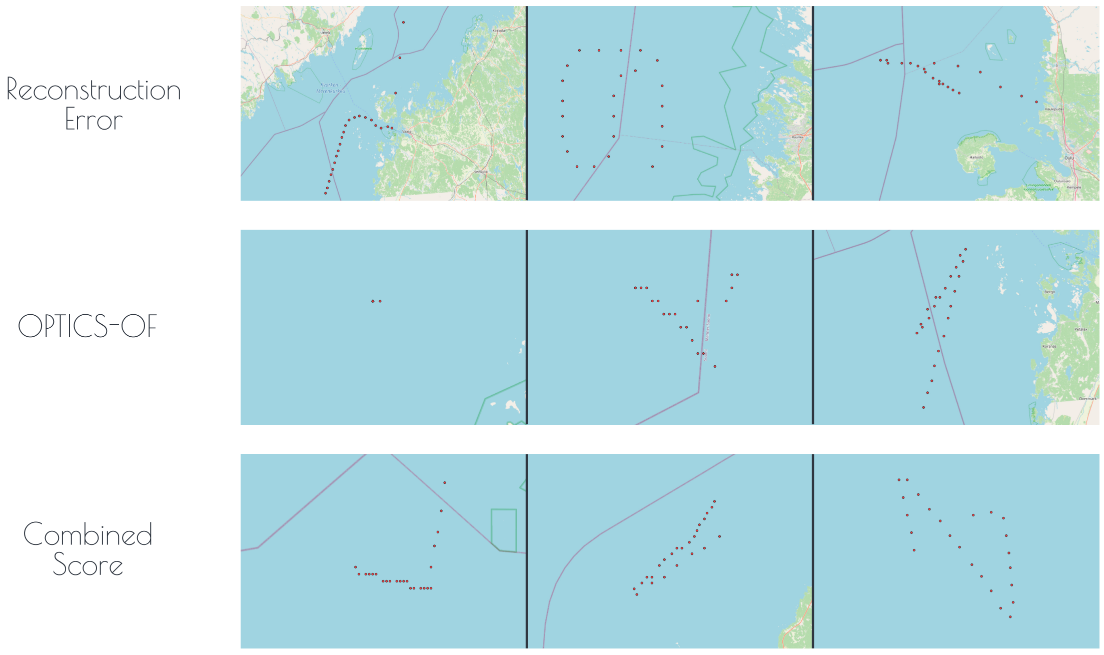

Figure 29: Example trajectories which the models trained with location-wise features assigned highest outlier scores. Three figures to best present the findings were selected for each outlier detection method.

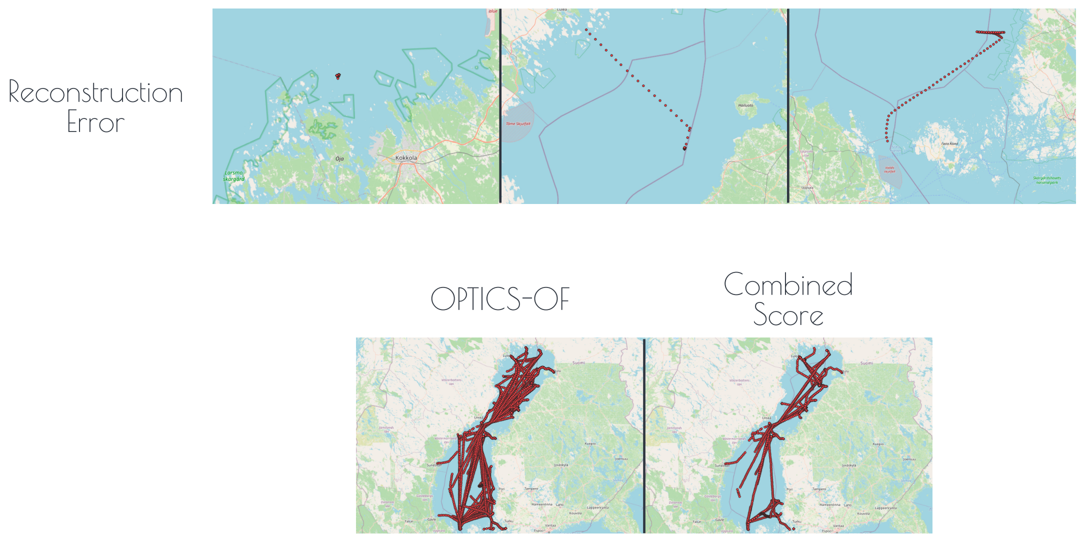

Figure 30: Example trajectories which the models trained with location-agnostic features assigned highest outlier scores. For reconstruction error, three figures to present the findings are displayed. For the OPTICS-OF and combined score, one image is displayed showing all trajectories with highest outlier scores.

9 Conclusions

The goal of this thesis was to evaluate the use of deep autoencoders for anomaly detection in maritime domain. The thesis proposes a new method which uses both reconstruction error and analysis of encodings by OPTICS-OF to construct an outlier score. The method and its components are evaluated on a benchmark Musk data set and a real-life AIS data set.

The results indicate that reconstruction error produced by an autoencoder can be used to identify anomalies as the data points with high reconstruction errors. The effectiveness of reconstruction error is dependent on the size of the encoding of an autoencoder and proportion of anomalies in the training set. The results show on the Musk data set that small encoding sizes work better for anomaly detection when using reconstruction error as an anomaly detection method. This supports the claim presented in [Agg17], which states that anomalies are more resistant to compression than non-anomalous data points in the context of a model trained on the data set. However, on the Musk data set results revealed a phenomenon where large encoding sizes increased performance in contradiction to the claim. Further research and analysis are required to identify the mechanics leading to the phenomenon.

The research on the thesis aimed to answer whether it was possible to increase the performance of anomaly detection by incorporating OPTICS-OF analysis of encodings to reconstruction error. OPTICS-OF was applied on the encodings produced by an autoencoder to produce outlier scores. The scores produced by OPTICS-OF and reconstruction error were then combined to form a combined outlier score. The results show improvement over reconstruction error in some cases, but in most cases, there was either no change and in some cases a decrease in performance. In summary, it can be stated that there is potential to improve performance by incorporating OPTICS-OF analysis of encodings, but more research is required to find perhaps a better function for the combination than the max function used in this thesis. The method should also be applied to a larger number of both benchmark and real-life data sets to draw further conclusions.

The behaviour of the OPTICS-OF was similar on all feature sets derived from both Musk and AIS data sets considering the number of large outlier scores assigned. The results also highlight that the choice of the parameter *MinPts* is crucial for the performance of the OPTICS-OF. On all feature sets, multiple values were tried for *MinPts*, and in both cases, the most relevant results were considered to be for the values where OPTICS-OF assigned fewer large outlier scores. Whether this phenomenon could be used as a way to identify good choices for the value of *MinPts* automatically requires further research.

A deep autoencoder and OPTICS-OF encoding analysis were applied on a real-life AIS data set containing ship trajectories with the goal of using the method to identify anomalous ship trajectories. Two distinct feature sets were derived from the raw AIS data. Reconstruction error successfully identified a large number of anomalous trajectories from both feature sets. OPTICS-OF was successful on other data set where it identified a set of anomalies distinct from the ones identified by reconstruction error. On the other data set, OPTICS-OF miss-classified too many cases to have been considered useful. In general, the findings of the method on the AIS data set show promising results and show potential for further research.

The results on Musk data set showed that there is a correlation of percentage anomalies in the training set and the performance of the trained model on anomaly detection task. The dependency between performance and the anomaly free training set provides a potential application for deep autoencoder and OPTICS-OF encoding analysis to be used to both identify anomalies in a data set and also be used to clean training data from anomalies.

A Musk Data Numeric Results

Table 1: Architecture Analysis: Decoder Size Comparisons on Autoencoder with Encoding Size 8 on Musk Data Set

Decoder Size	**Test MSE**	**OPTICS-OF Precision**	**Reconstruction Error Precision**
output	0.015	0.196	0.897
output + 32	0.008	0.196	0.639
output + 64	0.007	0.175	0.546
output + 96	0.007	0.170	0.784
output + 252	0.005	0.149	0.711

Table 2: Training Losses on Musk Data Set

Anomalies Train/Test	**Encoding**	**Train MSE**	**Test MSE**
0% / 100%	3	0.022	0.030
	6	0.016	0.025
	8	0.013	0.023
	12	0.009	0.018
	16	0.007	0.017
	24	0.004	0.012
	48	0.002	0.008
	96	0.001	0.006
40% / 60%	3	0.023	0.027
	6	0.016	0.019
	8	0.013	0.015
	12	0.009	0.010
	16	0.007	0.008
	24	0.004	0.005
	48	0.001	0.002
	96	0.001	0.001
80% / 20%	3	0.024	0.024
	6	0.017	0.017

	8	0.014	0.014
	12	0.009	0.010
	16	0.007	0.007
	24	0.004	0.004
	48	0.002	0.002
	96	0.001	0.001
100% / 0%	3	0.024	0.023
	6	0.017	0.017
	8	0.014	0.014
	12	0.009	0.009
	16	0.007	0.007
	24	0.004	0.004
	48	0.002	0.002
	96	0.001	0.001

Table 3: Reconstruction Error Precision on Musk Data Set

Anomalies Train/Test	**Encoding**	**Precision**
0% / 100%	3	1.000
	6	1.000
	8	1.000
	12	1.000
	16	1.000
	24	1.000
	48	1.000
	96	1.000
40% / 60%	3	1.000
	6	1.000
	8	0.876
	12	0.536
	16	0.649
	24	0.845
	48	1.000

	96	0.918
80% / 20%	3	1.000
	6	0.052
	8	0.052
	12	0.278
	16	0.361
	24	0.577
	48	0.897
	96	0.588
100% / 0%	3	0.835
	6	0.031
	8	0.052
	12	0.340
	16	0.258
	24	0.608
	48	0.649
	96	0.649

Table 4: OPTICS-OF Precisions on Musk Data Set

Encoding	**MinPts**	**0%**	**40%**	**80%**	**100%**
3	1	0.113	0.113	0.124	0.134
	2	0.113	0.258	0.196	0.278
	3	0.062	0.186	0.144	0.103
	4	0.175	0.196	0.155	0.165
	5	0.165	0.062	0.247	0.196
	7	0.144	0.093	0.124	0.124
	14	0.155	0.000	0.062	0.000
6	1	0.144	0.124	0.134	0.103
	2	0.227	0.227	0.155	0.216
	3	0.103	0.093	0.093	0.062
	4	0.155	0.124	0.165	0.103
	5	0.155	0.124	0.237	0.227
	7	0.206	0.134	0.103	0.155

	14	0.237	0.052	0.052	0.041
8	1	0.124	0.124	0.124	0.144
	2	0.196	0.237	0.247	0.237
	3	0.144	0.113	0.124	0.186
	4	0.186	0.134	0.165	0.278
	5	0.144	0.258	0.134	0.247
	7	0.124	0.134	0.113	0.134
	14	0.103	0.052	0.031	0.031
12	1	0.093	0.124	0.103	0.113
	2	0.237	0.258	0.247	0.247
	3	0.103	0.237	0.206	0.144
	4	0.196	0.186	0.196	0.144
	5	0.206	0.155	0.175	0.134
	7	0.165	0.113	0.124	0.144
	14	0.175	0.031	0.031	0.031
16	1	0.124	0.103	0.113	0.124
	2	0.237	0.247	0.227	0.227
	3	0.186	0.227	0.093	0.206
	4	0.227	0.196	0.155	0.196
	5	0.144	0.216	0.165	0.237
	7	0.124	0.175	0.144	0.227
	14	0.041	0.031	0.031	0.031
24	1	0.134	0.144	0.113	0.113
	2	0.196	0.247	0.216	0.227
	3	0.237	0.289	0.299	0.216
	4	0.237	0.309	0.289	0.289
	5	0.155	0.237	0.289	0.196
	7	0.093	0.206	0.124	0.113
	14	0.041	0.031	0.031	0.031
48	1	0.124	0.124	0.124	0.134
	2	0.227	0.216	0.216	0.258
	3	0.237	0.165	0.134	0.175
	4	0.196	0.155	0.155	0.206
	5	0.155	0.206	0.175	0.113
	7	0.072	0.124	0.082	0.082

	14	0.052	0.031	0.062	0.062
96	1	0.134	0.134	0.155	0.134
	2	0.216	0.278	0.278	0.258
	3	0.165	0.206	0.186	0.186
	4	0.216	0.186	0.186	0.175
	5	0.155	0.134	0.134	0.155
	7	0.082	0.062	0.124	0.196
	14	0.062	0.093	0.093	0.103

Table 5: Combined Outlier Score Precisions on Musk Data Set

Encoding	**MinPts**	**0%**	**40%**	**80%**	**100%**
3	1	0.990	1.000	1.000	0.845
	2	1.000	0.990	1.000	0.845
	3	1.000	0.990	1.000	0.835
	4	1.000	0.979	1.000	0.835
	5	1.000	0.979	1.000	0.835
	7	0.928	0.959	1.000	0.835
	14	0.918	0.918	0.804	0.825
6	1	1.000	1.000	0.062	0.041
	2	1.000	1.000	0.093	0.082
	3	0.876	0.959	0.082	0.093
	4	0.897	1.000	0.113	0.134
	5	0.897	0.990	0.165	0.186
	7	0.969	1.000	0.124	0.103
	14	0.918	0.897	0.062	0.041
8	1	0.990	0.876	0.093	0.082
	2	0.990	0.876	0.113	0.103
	3	0.918	0.835	0.072	0.072
	4	0.948	0.856	0.093	0.103
	5	0.969	0.856	0.113	0.062
	7	0.928	0.866	0.082	0.103
	14	0.825	0.670	0.031	0.031

12	1	0.990	0.536	0.289	0.361
	2	0.990	0.577	0.309	0.371
	3	0.938	0.505	0.309	0.351
	4	0.918	0.526	0.330	0.330
	5	0.959	0.526	0.320	0.340
	7	0.969	0.546	0.299	0.340
	14	0.938	0.474	0.175	0.320
16	1	1.000	0.660	0.361	0.268
	2	1.000	0.660	0.361	0.320
	3	0.959	0.680	0.351	0.258
	4	0.990	0.660	0.330	0.278
	5	0.938	0.660	0.330	0.278
	7	0.969	0.660	0.351	0.278
	14	0.928	0.567	0.268	0.227
24	1	1.000	0.845	0.588	0.608
	2	1.000	0.845	0.588	0.608
	3	0.979	0.835	0.577	0.598
	4	0.979	0.835	0.608	0.598
	5	1.000	0.835	0.577	0.588
	7	1.000	0.845	0.577	0.598
	14	0.990	0.629	0.577	0.536
48	1	1.000	1.000	0.897	0.649
	2	1.000	1.000	0.897	0.660
	3	1.000	0.948	0.825	0.649
	4	0.990	0.928	0.794	0.598
	5	0.990	0.979	0.845	0.629
	7	0.990	0.928	0.722	0.526
	14	0.948	0.845	0.639	0.526
96	1	1.000	0.928	0.598	0.649
	2	1.000	0.938	0.629	0.680
	3	1.000	0.907	0.588	0.701
	4	1.000	0.866	0.608	0.711
	5	1.000	0.918	0.619	0.670
	7	0.990	0.670	0.485	0.485

	14	0.918	0.546	0.474	0.495

B AIS Data Location Wise Features Numeric Results

Table 6: Training Losses on Location Wise Features

Encoding	**Train**	**Test**
2	0.005 341	0.005 687
4	0.001 386	0.001 560
8	0.000 371	0.000 381
16	0.000 422	0.000 436
24	0.000 389	0.000 401
48	0.000 371	0.000 381
96	0.000 335	0.000 342

Table 7: Reconstruction Control Data on Location Wise Features

Encoding	**0**	**1**	**2**	**3**	**4**	**5**
2	0.133	0.450	0.587	0.509	0.223	0.158
4	0.018	0.142	0.130	0.057	0.037	0.032
8	0.001	0.850	0.744	0.585	0.364	0.185
16	0.001	0.862	0.692	0.611	0.381	0.287
24	0.001	0.819	0.669	0.562	0.366	0.350
48	0.002	0.825	0.717	0.706	0.440	0.415
96	0.003	0.789	0.720	0.542	0.355	0.485

Table 8: OPTICS-OF on AIS Data Location Wise Features

Encoding	**MinPts**	**0**	**1**	**2**	**3**	**4**	**5**
	1	0.372	0.175	0.372	0.180	0.225	0.016
	2	0.948	0.445	0.548	0.065	0.075	0.015
	3	0.378	0.197	0.397	0.087	0.050	0.018
2							

	4	0.203	0.148	0.366	0.076	0.124	0.002
	5	0.164	0.151	0.562	0.069	0.124	0.002
	7	0.178	0.335	0.929	0.106	0.097	0.026
	14	0.220	0.425	0.297	0.668	0.027	0.933
	1	0.038	0.153	0.137	0.024	0.192	0.425
	2	0.034	0.197	0.159	0.025	0.017	0.038
	3	0.034	0.124	0.164	0.040	0.022	0.052
4	4	0.037	0.084	0.156	0.042	0.025	0.032
	5	0.039	0.085	0.190	0.050	0.026	0.034
	7	0.049	0.089	0.194	0.532	0.065	0.033
	14	0.124	0.180	0.332	0.256	0.120	0.079
	1	0.125	0.452	0.452	0.065	0.010	0.452
	2	0.039	0.510	0.601	0.562	0.016	0.140
	3	0.040	0.443	0.350	0.362	0.018	0.112
8	4	0.057	0.386	0.320	0.423	0.000	0.135
	5	0.089	0.322	0.283	0.283	0.023	0.176
	7	0.102	0.339	0.261	0.210	0.032	0.166
	14	0.081	0.204	0.055	0.045	0.040	0.175
	1	0.025	0.474	0.474	0.072	0.117	0.286
	2	0.020	0.527	0.527	0.272	0.138	0.099
	3	0.023	0.475	0.401	0.312	0.122	0.115
16	4	0.028	0.423	0.370	0.402	0.131	0.076
	5	0.037	0.402	0.446	0.407	0.105	0.104
	7	0.045	0.362	0.412	0.241	0.095	0.104
	14	0.059	0.343	0.315	0.049	0.046	0.153
	1	0.194	0.487	0.487	0.081	0.033	0.270
	2	0.024	0.491	0.689	0.491	0.040	0.157
	3	0.028	0.566	0.326	0.671	0.113	0.199
24	4	0.032	0.462	0.409	0.513	0.155	0.228
	5	0.038	0.449	0.423	0.354	0.184	0.179
	7	0.043	0.334	0.086	0.317	0.135	0.155
	14	0.056	0.287	0.048	0.048	0.059	0.178
	1	0.083	0.489	0.489	0.100	0.102	0.188
	2	0.027	0.582	0.780	0.532	0.041	0.121
	3	0.040	0.504	0.381	0.381	0.142	0.061

48

	4	0.052	0.459	0.403	0.259	0.204	0.049
	5	0.053	0.393	0.369	0.276	0.130	0.071
	7	0.051	0.339	0.087	0.054	0.102	0.071
	14	0.055	0.290	0.049	0.047	0.057	0.122
96	1	0.119	0.500	0.500	0.098	0.301	0.111
	2	0.030	0.428	0.428	0.299	0.130	0.028
	3	0.043	0.528	0.397	0.373	0.127	0.026
	4	0.049	0.426	0.320	0.272	0.152	0.038
	5	0.052	0.371	0.299	0.033	0.112	0.042
	7	0.051	0.311	0.294	0.049	0.092	0.042
	14	0.053	0.315	0.055	0.050	0.050	0.044

Table 9: Combined Score on AIS Data Location Wise Features

Encoding	**MinPts**	**0**	**1**	**2**	**3**	**4**	**5**
2	1	0.133	0.468	0.594	0.471	0.224	0.159
	2	0.133	0.635	0.703	0.396	0.224	0.159
	3	0.134	0.472	0.598	0.408	0.225	0.159
	4	0.134	0.442	0.578	0.402	0.225	0.136
	5	0.134	0.443	0.706	0.398	0.225	0.159
	7	0.134	0.558	0.955	0.420	0.226	0.159
	14	0.134	0.602	0.519	0.767	0.225	0.159
4	1	0.019	0.144	0.131	0.058	0.038	0.033
	2	0.018	0.143	0.131	0.058	0.037	0.032
	3	0.018	0.144	0.131	0.058	0.038	0.033
	4	0.019	0.144	0.132	0.058	0.039	0.033
	5	0.019	0.144	0.132	0.058	0.039	0.033
	7	0.019	0.145	0.133	0.059	0.039	0.033
	14	0.019	0.144	0.132	0.059	0.039	0.033
8	1	0.001	0.691	0.691	0.504	0.370	0.186
	2	0.001	0.720	0.765	0.745	0.368	0.186
	3	0.002	0.690	0.643	0.648	0.368	0.187
	4	0.002	0.661	0.632	0.680	0.000	0.188

	5	0.002	0.632	0.614	0.614	0.369	0.188
	7	0.002	0.641	0.604	0.583	0.368	0.187
	14	0.002	0.584	0.519	0.514	0.384	0.190
16	1	0.001	0.690	0.690	0.477	0.387	0.289
	2	0.001	0.715	0.715	0.578	0.388	0.289
	3	0.001	0.688	0.648	0.602	0.388	0.289
	4	0.001	0.656	0.630	0.645	0.390	0.290
	5	0.001	0.648	0.672	0.650	0.391	0.290
	7	0.001	0.627	0.654	0.569	0.392	0.290
	14	0.002	0.613	0.600	0.486	0.399	0.293
24	1	0.001	0.690	0.690	0.470	0.374	0.356
	2	0.002	0.689	0.799	0.689	0.378	0.360
	3	0.002	0.729	0.596	0.788	0.377	0.359
	4	0.002	0.671	0.642	0.698	0.380	0.361
	5	0.002	0.666	0.653	0.615	0.381	0.361
	7	0.002	0.602	0.486	0.595	0.382	0.362
	14	0.002	0.575	0.476	0.476	0.384	0.367
48	1	0.003	0.716	0.716	0.529	0.449	0.423
	2	0.002	0.759	0.894	0.733	0.452	0.424
	3	0.003	0.721	0.663	0.663	0.456	0.428
	4	0.004	0.703	0.675	0.610	0.461	0.433
	5	0.003	0.671	0.660	0.620	0.464	0.435
	7	0.003	0.647	0.541	0.528	0.465	0.434
	14	0.003	0.626	0.531	0.529	0.465	0.435
96	1	0.004	0.713	0.713	0.517	0.359	0.523
	2	0.004	0.679	0.679	0.618	0.363	0.493
	3	0.004	0.731	0.667	0.656	0.364	0.493
	4	0.005	0.683	0.632	0.609	0.367	0.503
	5	0.005	0.656	0.623	0.502	0.366	0.507
	7	0.004	0.629	0.621	0.511	0.364	0.508
	14	0.004	0.627	0.519	0.516	0.369	0.511

C AIS Data Location Agnostic Features Numeric Results

Table 10: Training Losses on Location Agnostic Features

Encoding	Train	Test
2	0.000 864	0.000 938
4	0.000 676	0.000 735
8	0.000 518	0.000 577
16	0.032 726	0.032 934
24	0.016 251	0.016 384
48	0.000 481	0.000 538
96	0.016 197	0.016 327

Table 11: Reconstruction Control Data on Location Agnostic Features

Encoding	0	1	2	3	4	5	6
2	0.006	0.602	0.304	0.294	0.224	0.728	0.044
4	0.000	0.632	0.493	0.534	0.262	0.888	0.062
8	0.005	0.582	0.565	0.613	0.281	0.749	0.056
16	0.011	0.514	0.490	0.533	0.385	0.609	0.100
24	0.005	0.501	0.524	0.571	0.367	0.622	0.098
48	0.005	0.527	0.506	0.565	0.244	0.667	0.054
96	0.006	0.496	0.539	0.560	0.349	0.619	0.101

Table 12: OPTICS-OF on AIS Data Location Agnostic Features

Encoding	MinPts	0	1	2	3	4	5	6
	1	0.405	0.405	0.405	0.405	0.405	0.405	0.361
	2	0.560	0.613	0.915	0.544	0.518	0.260	0.587
	3	0.980	0.692	0.584	0.800	0.592	0.299	0.453
2								

	4	0.173	0.165	0.565	0.740	0.321	0.297	0.248
	5	0.057	0.160	0.846	0.542	0.471	0.305	0.215
	7	0.049	0.097	0.772	0.562	0.400	0.442	0.151
	14	0.037	0.047	0.529	0.563	0.415	0.919	0.121
	1	0.404	0.404	0.158	0.153	0.056	0.081	0.215
	2	0.252	0.016	0.078	0.786	0.187	0.743	0.243
	3	0.874	0.016	0.094	0.943	0.933	0.664	0.247
4	4	0.670	0.007	0.130	0.981	0.325	0.420	0.188
	5	0.417	0.003	0.243	0.985	0.458	0.594	0.143
	7	0.018	0.005	0.411	0.941	0.259	0.936	0.105
	14	0.001	0.025	0.364	0.798	0.121	0.778	0.136
	1	0.404	0.341	0.042	0.423	0.423	0.107	0.037
	2	0.982	0.490	0.599	0.006	0.841	0.237	0.030
	3	0.979	0.606	0.441	0.030	0.760	0.306	0.035
8	4	0.931	0.527	0.453	0.134	0.996	0.164	0.060
	5	0.882	0.243	0.468	0.175	0.992	0.120	0.078
	7	0.082	0.083	0.457	0.544	0.881	0.090	0.123
	14	0.002	0.073	0.629	0.760	0.653	0.051	0.121
	1	0.420	0.420	0.420	0.420	0.420	0.420	0.065
	2	0.907	0.638	0.892	0.470	0.454	0.539	0.120
	3	0.839	0.496	0.030	0.992	0.715	0.925	0.191
16	4	0.987	0.272	0.030	0.608	0.705	0.240	0.280
	5	0.787	0.258	0.074	0.140	0.766	0.188	0.441
	7	0.527	0.116	0.140	0.128	0.444	0.155	0.428
	14	0.273	0.054	0.825	0.805	0.536	0.448	0.381
	1	0.422	0.422	0.422	0.422	0.422	0.012	0.019
	2	0.782	0.147	0.656	0.795	0.643	0.019	0.043
	3	0.748	0.156	0.936	0.629	0.446	0.047	0.165
24	4	0.363	0.085	0.297	0.941	0.536	0.127	0.204
	5	0.035	0.109	0.168	0.581	0.395	0.204	0.303
	7	0.004	0.146	0.209	0.232	0.363	0.550	0.319
	14	0.001	0.177	0.788	0.691	0.592	0.408	0.272
	1	0.435	0.170	0.435	0.435	0.435	0.435	0.137
	2	0.959	0.598	0.014	0.803	0.426	0.672	0.060
	3	0.761	0.765	0.007	0.628	0.787	0.797	0.156

48

	4	0.785	0.750	0.092	0.454	0.633	0.331	0.161
	5	0.696	0.408	0.119	0.531	0.496	0.414	0.177
	7	0.695	0.270	0.647	0.623	0.487	0.681	0.215
	14	0.388	0.040	0.765	0.824	0.363	0.721	0.484
96	1	0.137	0.295	0.455	0.455	0.455	0.455	0.023
	2	0.692	0.120	0.302	0.967	0.902	0.835	0.032
	3	0.526	0.065	0.010	0.966	0.880	0.904	0.063
	4	0.717	0.051	0.030	0.614	0.622	0.888	0.210
	5	0.053	0.047	0.020	0.586	0.599	0.562	0.181
	7	0.026	0.048	0.064	0.568	0.556	0.552	0.200
	14	0.010	0.122	0.213	0.846	0.399	0.597	0.206

Table 13: Combined Score on AIS Data Location Agnostic Features

Encoding	**MinPts**	**0**	**1**	**2**	**3**	**4**	**5**	**6**
2	1	0.006	0.620	0.307	0.297	0.227	0.620	0.045
	2	0.006	0.751	0.304	0.294	0.224	0.525	0.044
	3	0.006	0.820	0.422	0.405	0.286	0.555	0.049
	4	0.006	0.409	0.415	0.397	0.280	0.550	0.048
	5	0.006	0.401	0.413	0.396	0.283	0.556	0.048
	7	0.007	0.271	0.423	0.409	0.305	0.659	0.057
	14	0.009	0.139	0.411	0.399	0.307	0.954	0.064
4	1	0.001	0.603	0.440	0.438	0.264	0.391	0.062
	2	0.001	0.083	0.247	0.873	0.367	0.846	0.074
	3	0.001	0.094	0.285	0.966	0.358	0.791	0.071
	4	0.001	0.039	0.336	0.988	0.361	0.626	0.074
	5	0.001	0.018	0.485	0.990	0.362	0.745	0.078
	7	0.001	0.037	0.607	0.960	0.342	0.957	0.071
	14	0.001	0.139	0.559	0.865	0.324	0.851	0.072
8	1	0.006	0.551	0.348	0.610	0.283	0.390	0.058
	2	0.006	0.670	0.750	0.043	0.397	0.467	0.065
	3	0.006	0.757	0.635	0.142	0.397	0.526	0.066
	4	0.006	0.699	0.642	0.340	0.389	0.383	0.065

	5	0.006	0.468	0.650	0.403	0.376	0.333	0.063
	7	0.006	0.272	0.645	0.709	0.374	0.283	0.062
	14	0.006	0.239	0.781	0.866	0.380	0.191	0.064
16	1	0.011	0.593	0.593	0.593	0.593	0.593	0.100
	2	0.011	0.745	0.922	0.623	0.613	0.672	0.105
	3	0.011	0.640	0.269	0.994	0.799	0.947	0.106
	4	0.011	0.472	0.263	0.718	0.788	0.449	0.107
	5	0.011	0.458	0.317	0.371	0.832	0.407	0.108
	7	0.011	0.347	0.367	0.356	0.592	0.379	0.108
	14	0.012	0.284	0.880	0.865	0.658	0.589	0.111
24	1	0.005	0.593	0.593	0.593	0.593	0.303	0.098
	2	0.006	0.381	0.764	0.860	0.754	0.165	0.104
	3	0.006	0.386	0.955	0.737	0.606	0.290	0.102
	4	0.006	0.327	0.487	0.960	0.669	0.362	0.101
	5	0.005	0.344	0.390	0.698	0.560	0.417	0.102
	7	0.005	0.368	0.419	0.436	0.535	0.679	0.102
	14	0.006	0.392	0.860	0.791	0.718	0.576	0.105
48	1	0.005	0.429	0.617	0.617	0.245	0.617	0.054
	2	0.005	0.744	0.099	0.881	0.324	0.796	0.060
	3	0.005	0.860	0.039	0.772	0.337	0.881	0.065
	4	0.005	0.854	0.262	0.651	0.341	0.551	0.067
	5	0.005	0.616	0.311	0.708	0.342	0.621	0.069
	7	0.006	0.494	0.786	0.770	0.345	0.809	0.072
	14	0.005	0.202	0.849	0.890	0.304	0.819	0.057
96	1	0.006	0.496	0.615	0.615	0.615	0.615	0.101
	2	0.007	0.357	0.499	0.976	0.930	0.882	0.107
	3	0.007	0.305	0.110	0.977	0.918	0.933	0.109
	4	0.007	0.288	0.259	0.728	0.735	0.924	0.110
	5	0.007	0.275	0.150	0.709	0.719	0.691	0.111
	7	0.007	0.276	0.298	0.698	0.688	0.686	0.111
	14	0.007	0.354	0.429	0.897	0.574	0.723	0.108

References

[AAB$^+$15] Abadi, Martín, Agarwal, Ashish, Barham, Paul, *et al.*: *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015. `http://tensorflow.org/`, Software available from tensorflow.org.

[ABKS99] Ankerst, Mihael, Breunig, Markus M., Kriegel, Hans Peter, and Sander, Jörg: *Optics: Ordering points to identify the clustering structure*. SIGMOD Rec., 28(2):49–60, June 1999, ISSN 0163-5808. `https://doi.org/10.1145/304181.304187`.

[AC15] An, Jinwon and Cho, Sungzoon: *Variational autoencoder based anomaly detection using reconstruction probability*. Special Lecture on IE, 2(1):1–18, 2015.

[Agg17] Aggarwal, Charu C.: *Outlier Analysis*. Springer International Publishing AG, 2017.

[AS15] Aggarwal, Charu C. and Sathe, Saket: *Theoretical foundations and algorithms for outlier ensembles*. SIGKDD Explor. Newsl., 17(1):24–47, September 2015, ISSN 1931-0145. `https://doi.org/10.1145/2830544.2830549`.

[BCMLK16] Bontemps, Loïc, Cao, Van Loi, McDermott, James, and Le-Khac, Nhien An: *Collective anomaly detection based on long short-term memory recurrent neural networks*. In Dang, Tran Khanh, Wagner, Roland, Küng, Josef, Thoai, Nam, Takizawa, Makoto, and Neuhold, Erich (editors): *Future Data and Security Engineering*, pages 141–152, Cham, 2016. Springer International Publishing, ISBN 978-3-319-48057-2.

[BKNS99] Breunig, Markus M., Kriegel, Hans Peter, Ng, Raymond T., and Sander, Jörg: *Optics-of: Identifying local outliers*. In Żytkow, Jan M. and

Rauch, Jan (editors): *Principles of Data Mining and Knowledge Discovery*, pages 262–270, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg, ISBN 978-3-540-48247-5.

[Bom06] *Associative Learning of Vessel Motion Patterns for Maritime Situation Awareness*, July 2006, ISBN 1-4244-0953-5.

[Bra16] *Computer age statistical inference Algorithms, Evidence, and Data Science.* Cambridge University Press, 2016.

[Cla94] Clarkson, Kenneth L.: *An algorithm for approximate closest-point queries.* In *Proceedings of the Tenth Annual Symposium on Computational Geometry*, SCG '94, page 160–164, New York, NY, USA, 1994. Association for Computing Machinery, ISBN 0897916484. `https://doi.org/10.1145/177424.177609`.

[CVMG$^+$14] Cho, Kyunghyun, Van Merriënboer, Bart, Gulcehre, Caglar, Bahdanau, Dzmitry, Bougares, Fethi, Schwenk, Holger, and Bengio, Yoshua: *Learning phrase representations using rnn encoder-decoder for statistical machine translation.* arXiv preprint arXiv:1406.1078, 2014.

[GaJ15] *An Introduction to Statistical Learning with Applications in R.* Springer Science+Business Media New York, 2015.

[GBC16] Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron: *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.

[GMfMPIfMitS14] Sciences, Razvan Pascanu from Universite de Montreal, Kyunghyun Cho from Universite de Montreal Yoshua Bengio from Universite de Montreal Guido

Montufar from Max Planck Institute for Mathematics in the: *Ont he number of linear regions of deep neural networks*, 2014.

[HHS17] Hoffer, Elad, Hubara, Itay, and Soudry, Daniel: *Train longer, generalize better: closing the generalization gap in large batch training of neural networks.* In *NIPS*, 2017.

[HS13] Hermans, Michiel and Schrauwen, Benjamin: *Training and analysing deep recurrent neural networks.* In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q. (editors): *Advances in Neural Information Processing Systems 26*, pages 190–198. Curran Associates, Inc., 2013. `http://papers.nips.cc/paper/5166-training-and-analysing-deep-recurrent-neural-networks.pdf`.

[HZRS15] He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian: *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification.* In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[JZZX13] Jiang, Xiaojuan, Zhang, Yinghua, Zhang, Wensheng, and Xiao, Xian: *A novel sparse auto-encoder for deep unsupervised learning.* In *2013 Sixth International Conference on Advanced Computational Intelligence (ICACI)*, pages 256–261. IEEE, 2013.

[KB14] Kingma, Diederik P and Ba, Jimmy: *Adam: A method for stochastic optimization.* arXiv preprint arXiv:1412.6980, 2014.

[Kow12] *Maritime anomaly detection using Gaussian Process active learning*, January 2012, ISBN 978-1-4673-0417-7.

[LZW18] Li, Feng, Zuraday, JM, and Wu, Wei: *Sparse representation learning of data by autoencoders with l^ sub 1/2^ regularization.* Neural Network World, 28(2):133–147, 2018.

[ML20] Maija Luotonen, Helsingin Sanomat: *Hollantilainen rahtialus ajelehti kohti rannikkoa selkämerellä, laivan kone saatiin tuntien jälkeen kuntoon.* January 2020.

[NS16] Nanduri, Anvardh and Sherry, Lance: *Anomaly detection in aircraft data using recurrent neural networks (rnn).* pages 5C2–1, April 2016.

[NVH$^+$18] Nguyen, D., Vadaine, R., Hajduch, G., Garello, R., and 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 (DSAA)*, pages 331–340, Oct 2018.

[Pan14] *Introduction to Data Mining.* Pearson Education Limited, 2014.

[PeF12] *Machine Learning The Art and Science of Algorithms that Make Sense of Data.* Cambridge University Press, 2012.

[PeN10] *Artificial Intelligence A Modern Approach.* Pearson Education, Inc., publishing as Prentice Hall, 2010.

[PGC$^+$17] Paszke, Adam, Gross, Sam, Chintala, Soumith, Chanan, Gregory, Yang, Edward, DeVito, Zachary, Lin, Zeming, Desmaison, Alban, Antiga, Luca, and Lerer, Adam: *Automatic differentiation in pytorch.* 2017.

[PVD$^+$17] Protopapadakis, Eftychios, Voulodimos, Athanasios, Doulamis, Anastasios, Doulamis, Nikolaos, Dres, Dimitrios, and Bimpas, Matthaios: *Stacked autoencoders for outlier detection in over-the-horizon radar signals.* Computational intelligence and neuroscience, 2017, 2017.

[Ray16] Rayana, Shebuti: *ODDS library*, 2016. `http://odds.cs.stonybrook.edu`.

[RiS18] *Reinforcement Learning An Introduction.* The MIT Press, 2018.

[Ros58] Rosenblatt, Frank: *The perceptron: A probabilistic model for information storage and organization in the brain.* Psychological Review, 65:386–408, 1958.

[SKL17] Smith, Samuel L., Kindermans, Pieter-Jan, and Le, Quoc V.: *Don't decay the learning rate, increase the batch size.* CoRR, abs/1711.00489, 2017. `http://arxiv.org/abs/1711.00489`.

[YZZ+18] Yao, Di, Zhang, Chao, Zhu, Zhihua, Hu, Qin, Wang, Zheng, Huang, Jianhui, and Bi, Jingping: *Learning deep representation for trajectory clustering.* Expert Systems, 35(2):e12252, 2018. `https://onlinelibrary.wiley.com/doi/abs/10.1111/exsy.12252`, e12252 10.1111/exsy.12252.

[ZP17] Zhou, Chong and Paffenroth, Randy C: *Anomaly detection with robust deep autoencoders.* In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 665–674, 2017.