Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Cecilie Haugsbø Hermansen

Using Deep Learning for Anomaly Detection in Distributed Temperature Sensing Systems for Submarine Export Cables

Master's thesis in Computer Science
Supervisor: Keith Downing
June 2019

NTNU
Kunnskap for en bedre verden

Cecilie Haugsbø Hermansen

Using Deep Learning for Anomaly Detection in Distributed Temperature Sensing Systems for Submarine Export Cables

Master's thesis in Computer Science
Supervisor: Keith Downing
June 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

Export cables are essential components in offshore wind farms, as they are responsible for transporting the produced power to the shore. Any failures in the export cables can lead to huge losses for the operators, and thus they want to monitor the cables to be alerted of any developing faults that can be corrected. Physical laws govern the relationships of the temperatures of the cable, the electric current through the cable and the surrounding environment. These relationships are complicated to model physically, but might be implicitly learned by a machine learning model.

Asynchronous data sources describe the system of the export cable and its surroundings. Experiments were conducted to establish how best to handle this asynchronicity in deep learning prediction models. The preferred model was an LSTM with inputs resampled to the same uneven sampling rate.

No known failures have occurred in the export cables. The normal behaviour of the temperatures of the cable was modelled, and deviating behaviour was used to detect anomalies. Synthetic faults were constructed to evaluate the anomaly detection models.

Anomaly detection based on temporal and spatial relationships was explored separately. The best models were combined, but the combined model did not beat the best spatial model. The best model used principal component analysis to encode and reconstruct sequences of temperatures through the cable, and got impressive results for the constructed gradually developing faults.

Additionally, a new cross validation based evaluation scheme for time series data with seasonality is proposed. It involves dividing the data into small consecutive blocks, and randomly dividing the blocks into the required number of folds.

Sammendrag

(Abstract in Norwegian)

Eksportkabler er viktige komponenter i en offshorevindmøllepark, siden de transportere den produserte energien til land. Feil i kablene kan føre til store tap for operatørene, og de vil derfor overvåke kablene for å kunne varsle om feil under utvikling som kan korrigeres. Fysiske lover regulerer forholdet mellom temperaturene i eksportkablene, strømmen gjennom kabelen og omgivelsene. Det er komplisert å fysisk modellere disse forholdene, men de kan læres implisitt av en dyp læring-modell.

Asynkrone datakildene beskriver systemet rundt eksportkabelen. Ekperimenter ble utført for å fastslå hvordan asynkronisiteten best skulle håndteres i dyp læringmodeller. Den foretrukne modellen var en LSTM modell med inputdata resamplet til å være synkron, men ugjevnt samplet.

Det er ikke kjent at det har vært feil i kabelen. Den normale oppførselen til temperaturene i kabelen ble modellert, og avvik ble brukt til å finne feil. Syntetiske feil ble konstruert for å evaluere de forskjellige modellene. Feildeteksjon basert på temporale og spatiale forhold ble undersøkt hver for seg. De beste modellen ble kombinert, men slo ikke den beste spatiale modellen. Den beste modellen bruke prinsipalkomponentanalyse (principal component analysis, PCA) til å encode og rekonstruere sekvenser av kabeltemperaturer, og fikk gode resultater på de konstruerte feilene.

Til slutt er en ny metode for trening og validering av modeller for periodisk tidsseriedata foreslått. Den er basert på kryssvalidering, og går ut på å dele dataene opp i små, sammenhengende blokker som tilfeldig fordeles på det ønskede antallet deler.

Preface

This Masters thesis was written for the Department of Computer Science at NTNU, and supervised by Keith L. Downing. It was supported by the software innovation department of Equinor in Trondheim, with data, domain knowledge and computational resources. I would like to thank my Equinor supervisors Vidar Slåtten and Thomas Helfer for their continued support and enthusiasm for the project.

Cecilie Haugsbø Hermansen
Trondheim, June 11, 2019

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter provides the background and motivation of the thesis. Section 1.2 introduces the goal and research questions that have guided the research. The research method is described in section 1.3, and the contributions of the thesis are summarized in section 1.4. Section 1.5 presents the structure of the rest of the thesis.

1.1 Background and Motivation

In 2015, the UN launched 17 global sustainable development goals for 2030. The 13th goal concerns climate change and promotes developments in renewable energy [UN.org, 2019]:

> *Take urgent action to combat climate change and its impacts by regulating emissions and promoting developments in renewable energy.*

Wind power is a renewable and emission free source of electricity. A wind farm uses wind turbines to convert the kinetic energy of the wind into electrical power, and needs stable and strong wind conditions for stable power production. Wind turbines are large and prominent. As a consequence, building an onshore wind farm can lead to interest conflicts, where the demand for renewable energy is met with concern for interventions in nature. In building wind farms offshore, fewer conflicts of interest arise. Offshore installations can also get access to better and more stable wind resources, and there are large areas available for construction [vindportalen.no, 2019]. However, offshore wind farms are more expensive to build, operate and maintain. Thus it is important for operators reduce the costs where possible, for offshore wind to be a profitable and attractive investment.

Export cables are a crucial part of an offshore wind farm installation, since all the produced power has to pass through them. The failure of an export cable can

have huge effects, as it will lead to a significant decrease in the amount of power being transported from the wind farm. The operators want to keep the cables in good condition, and want to be notified about any measurements that can be considered anomalous. This could indicate that a cable is developing a problem, and further investigation can be carried out that might lead to finding and correcting faults before they evolve beyond easy correction. The current monitoring of the export cables consists of temperature threshold values based on the properties of the cable. When the temperature in a segment of the cable reaches a threshold value, an alarm sounds in the control room. These thresholds are crude values that warn of an occurring failure. They cannot detect gradually developing faults at an early stage, and thus do not facilitate correcting faults at an early stage.

A distributed temperature sensing (DTS) system indirectly monitors the temperatures at discrete points through the export cable. Physical laws govern the relationships between the electric current through the cable, the temperature of the cable and other surrounding variables. These physical relationships are complicated to model mathematically, as not enough data is available from the environment surrounding the cable. How the heat dissipates from the cable depends on the thermal conductivity of the surrounding material, which can include soils, clays and sea water depending on the burial location. The thermal conductivity varies along the cable, and the heat dissipation also depends on other factors such as the sea current and sea temperature. Since the heat dissipation cannot feasibly be mathematically modelled, this motivates the use of deep learning to instead implicitly learn the complicated physical relationships governing the cable and its environment.

The studied export cables have yet to experience any failures. There are however plentiful observations of the normal behaviour of the cables. This motivates focusing on unsupervised methods for anomaly detection.

1.2 Goals and Research Questions

The following overarching goal has guided this thesis:

Goal *Find the best deep learning approach to anomaly detection in the submarine export cables.*

To guide towards achieving this goal and limit the scope of the research, two research questions were constructed.

Research question 1 *How are the asynchronous time series of the export cable system best handled in sequential deep learning models?*

The measurements of the DTS system together with related variables make up an asynchronous time series. The asynchronicity must be handled when using the data for anomaly detection. Different ways to handle asynchronous time series when using deep learning for time series prediction are reviewed and compared. In particular, different resampling schemes are compared. Also including time information explicitly in the input features is attempted, and deep learning models specialized to handle asynchronous data are reviewed. It is hypothesized that models not depending on resampling the inputs give more accurate predictions.

Research question 2 *Does exploiting relationships in both the temporal and spatial dimensions of the export cable data give more robust anomaly detection than only regarding one of the dimensions?*

Each observation of the export cable has a spatial dimension, and observations over time give a temporal dimension. The information needed to find anomalies can be contained in the spatial distribution of the temperatures, in the temporal evolution of the temperature of a single segment, or distributed over both dimensions. The hypothesis is that using information from both dimensions makes for superior anomaly detection. It might however be easier to learn how best to exploit the information captured in each dimension, when regarding each dimension on its own. Combining the approaches that work best for each dimension into one model will hopefully give even better anomaly detection results.

Attempting to answer this research question includes finding the best deep learning approaches to anomaly detection using information only in the spatial dimension, only in the temporal dimension and in both dimensions.

1.3 Research Method

A structured literature review is carried out to establish the state of the art of anomaly detection in DTS systems, anomaly detection using deep learning and how asynchronous time series data is handled in deep learning. With the knowledge gained from this, practical experiments are devised to see how best to handle the asynchronous export cable data in deep learning prediction models. A second series of experiments investigates anomaly detection based on temporal information only, spatial information only and temporal and spatial information combined.

1.4 Contributions

The main contributions of the thesis are outlined here. They are further discussed in section 6.2. They include:

- A proposed complete anomaly detection scheme for the export cables. It is based on using principal component analysis to encode and reconstruct the sequence of DTS values at a single time step. The reconstruction error is used to differentiate normal and anomalous behavior.
- A proposed anomaly score for high-dimensional error vectors that can be used to detect faults of a local character.
- A proposed scheme for how to train and evaluate models with time series data with seasonality, using cross validation with small sequential blocks randomly distributed to the different folds.

1.5 Thesis Structure

The next chapter reviews the background theory deemed necessary to understand the contributions of the thesis. Chapter 3 presents an overview of related work on anomaly detection and asynchronous time series data in deep learning. It also summarizes the motivation of the thesis. Chapter 4 presents the experiments conducted on asynchronous time series in deep learning prediction models. The model architectures are detailed, the experimental setup is explained, and the results are presented and discussed. Chapter 5 builds on the results of its preceding chapter with experiments on anomaly detection in the export cables. First the experimental setup is explained. The chapter is further split into three main sections, where the temporal, spatial and combined experiments are described separately. In each of the sections, the model architectures are detailed, and results are presented and discussed. Chapter 6 wraps up the thesis with a final discussion of all the obtained results, a discussion of the contributions of the thesis and proposal of future work.

Chapter 2

Background Theory

This chapter reviews background theory useful for understanding the contributions of the thesis.

2.1 Export Cables and DTS Systems

Export cables contribute a significant capital expenditure in the construction of an offshore wind farm. The cables are required to have sufficient current carrying capacity, so that they are not damaged by high temperatures at high currents. Due to the high cost there is usually little redundancy in the cables, which makes the condition of the export cables critical. Each cable is buried in varying seabed materials and at varying depths. As the power output of the wind farm is linked to wind conditions, the output varies in ways difficult to predict and the cable is exposed to varying electric currents.

A distributed temperature sensing system (DTS system) is a technology used for export cables to indirectly monitor the temperatures of the cable. The technology provides temperature measurements along a fibre optic cable [Ukil et al., 2011]. The temperature is measured along the cable by having a short laser pulse emitted. As the light travels along the fiber, it collides with the atomic lattice structure of the side walls of the fibre and is reflected back to the source at slightly shifted frequencies. The mechanical properties of the side wall affect the scattered frequencies, and the shifts in frequencies can be used to detect strain in the fibres. The strain is influenced by temperature, pressure and direct force. The returning light is analyzed by a recording instrument that can determine the strain. Changes in strain are assumed to be influenced only by the temperature. It can however also be affected by the cable having to support itself due to changes of the supporting ground, such as the seabed being washed away from under the cable. Since light travels at constant speed, the two-way travel time since the pulse was emitted

determines the location of the measurements along the cable. In this manner a continuous temperature profile can be made available, but in practice the data is discretized during the analysis.

At Dudgeon offshore wind farm off the coast of Norfolk in UK, two export cables are responsible for exporting the generated power to the mainland. A map showing the wind farm and the approximate route of the export cables can be seen in figure 2.1. There is little to no redundancy in the current carrying capacity of the two cables at full production, making each an important asset. The distribution of the power between the two cables is explicitly controlled, such that each wind turbine generator is connected to only one of the cables at any time. The experiments of this thesis will only use data from export cable 1, to reduce the scope of the experiments.

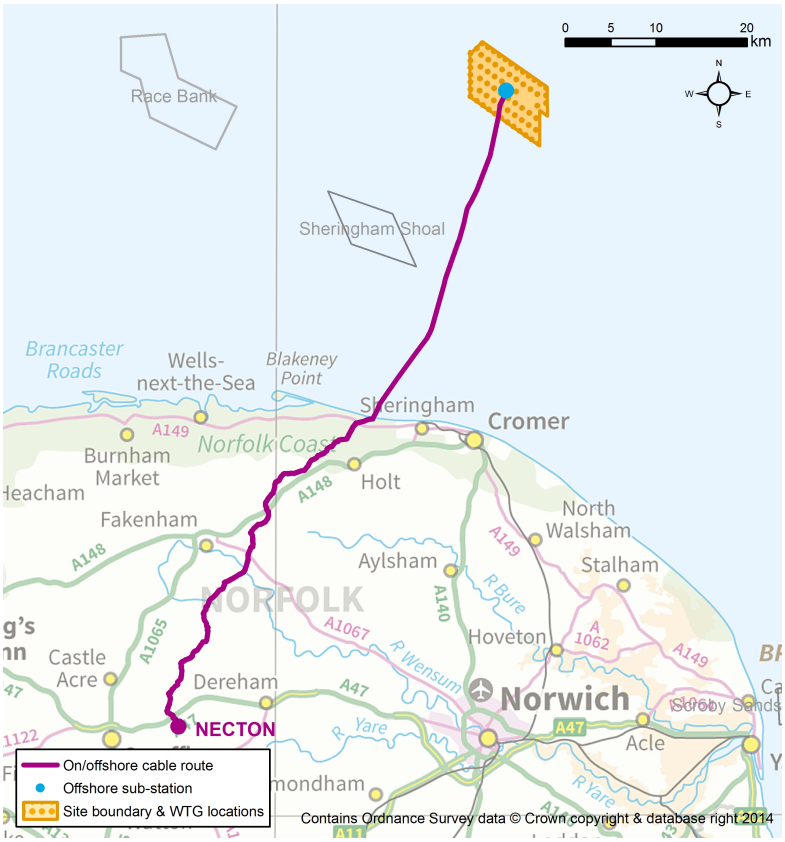


Figure 2.1: Map giving an overview of Dudgeon offshore wind farm. The drawing is simplified to only show one cable, while in reality there are two cables. The thesis is only concerned with the submarine portions of the cables. Image from [statkraft.no, 2019].

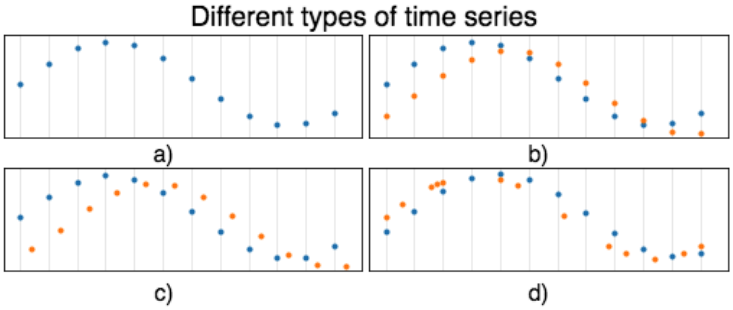


Figure 2.2: Some of the different classes of time series. a) Univariate time series with uniform samples. b) Synchronous multivariate time series with uniform samples. c) Asynchronous multivariate time series with uniform samples. d) Asynchronous multivariate time series with uneven samples.

2.2 Time Series Terminology

This section introduces the terminology used to describe the characteristics of a time series. A time series is a collection of ordered data points indexed by time, and is usually a repeated measurement of the same variables. Univariate time series consist of a single variable varying over time, while multivariate time series consist of multiple variables. These variables are often highly interrelated.

A time series can either be uniformly or unevenly sampled. Uniformly sampled time series have their data points equally spaced in time, while an unevenly sampled time series has unequally spaced data points. Multivariate time series can be either synchronous with all variables recorded at time same time, or asynchronous with variables recorded at different times. Figure 2.2 shows some of the different time series classes.

Methods for time series analysis often assume synchronous, uniformly sampled time series. Many real world applications produce time series of an asynchronous character. Methods exist for transforming time series to a uniformly sampled, synchronous format, but this comes with a cost. Often different resampling techniques are employed, inventing new values to make a time series uniformly and synchronously sampled. A signal can be upsampled to a higher frequency to match the sampling rate of other variables. Linear interpolation is often used to infer the new values. When downsampling a signal to a lower frequency, linear interpolation can again be used, but it is also possible to use aggregated values such as the average, sum or maximum value.

All types of resampling come with the cost of either loss of information or enlargement of the data set. Interpolation for upsampling results in both. Larger data sets make the analysis more computationally expensive and can hurt performance, and thus an enlargement of the data set is not desired.

2.3 Export Cable Data Sources

With the time series terminology in place, the data sources available to the experiments can be further introduced. The data available includes measurements from the DTS systems monitoring the two export cables at Dudgeon offshore wind farm. There are also observations of the electric current through each cable, and of the sea temperature at a central point in the wind farm.

DTS Data

The bulk of the data available stems from the DTS systems. For the cable used in the experiments, 39516 observations are available at each time step. Each individual observation is referred to as a segment of the cable, and the segments are approximately one meter apart in distance. The DTS data is unevenly sampled, with 50% of the sampling intervals between 16.5 and 21.4 minutes. 95% are between 14.6 and 24.4 minutes.

The DTS data can be indexed in two dimension; spatially and temporally. Defining $t(\tau, s)$ to be the DTS value at time step τ measured in segment s, the segment s can be varied to see how the DTS value at a given time τ varies along the cable. Figure 2.3 plots two snapshots of the DTS value profile of cable 1. Many of the same trends can be seen in the two snapshots, even if they are more than one year apart and of different magnitudes. The correlation between the sequences is 0.91. This hints at strong spatial relationships that hopefully are possible for a deep learning model to learn.

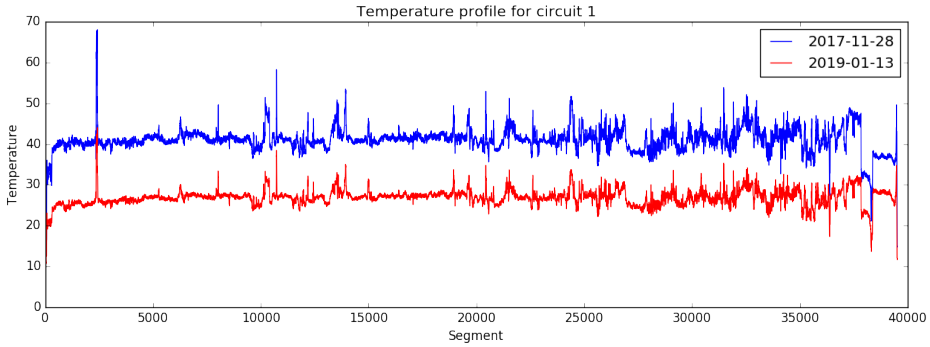


Figure 2.3: Snapshots of the DTS values of the cable

In figure 2.4 the segment s is frozen and the time τ varied, so that two distinct segments are plotted for the same time period. The cable segments in question are located at 70.141 and 3996.621 meter from the shore for the orange and blue signals respectively. Although some of the same trends are present for both segments, the

relationship between the segments does not seem to be trivial. The correlation between the plotted sequences is 0.85.

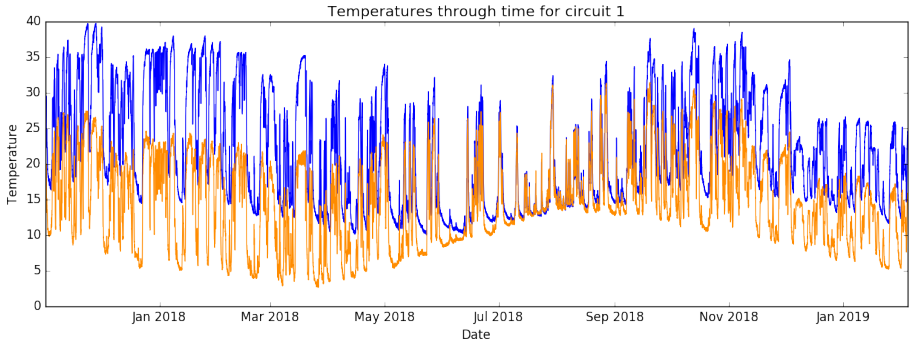


Figure 2.4: The DTS values of two cable segments through time

Further analysis of the correlation between the segments also hints at strong spatial correlation. In figure 2.5, the correlation matrix of the temperatures of the first 500 segments is plotted. Big patches of dark red color show big groups of highly correlated temperatures. This might hint that dimensionality reduction techniques should be able to successfully reduce the temperature profile to a much smaller dimension.

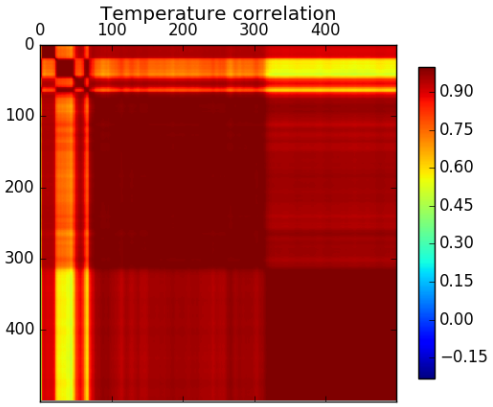


Figure 2.5: DTS correlation matrix for first 500 segments.

Electric Current

The data source with the highest frequency is the electric current through the cable. The univariate time series is uniformly sampled at 10 minute intervals. The plots in figure 2.6 show the temperature in a random cable segment and the current

through the cable in the same time period. From inspection of the plots, it can be seen that the temperature trend is a somewhat rounded version of the current trend. The electric current in believed to be the biggest driver of the evolution of the temperatures of the cable.

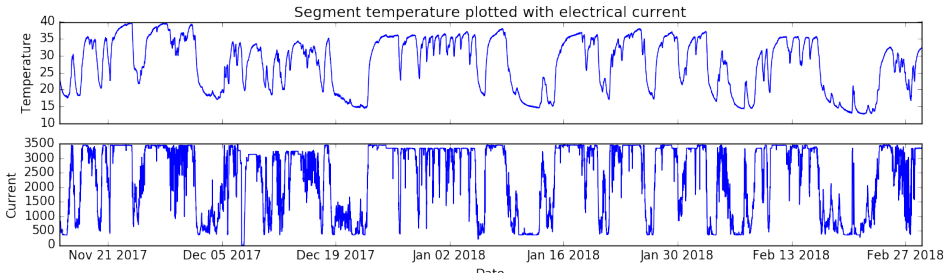


Figure 2.6: The DTS values of a single segment plotted with the electric current.

Sea Temperature

Only a centralised sea temperature reading from Dudgeon offshore wind farm was available, not sea temperature values along the cable route. The sea temperature is plotted in figure 2.7. It shows a seasonal character, with the temperature increasing from March to August, before decreasing again from August to March. The data of the sea temperature will not accurately describe the sea temperature along the different segments of the cable, but might serve as a proxy for the overall trend. The sea temperature data is uniformly sampled at every whole hour.

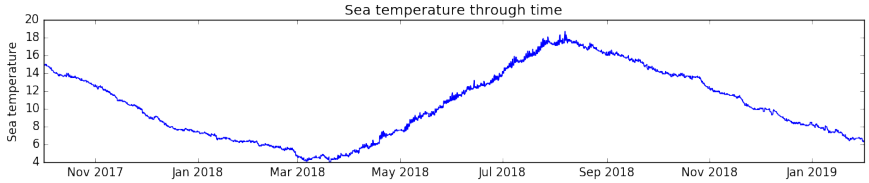


Figure 2.7: Sea temperature measured at a central point in Dudgeon offshore wind farm.

Combining the Data Sources

The electric current and sea temperature are believed to be important drivers of the evolution of the temperature in the cable. This motivates joining the three data sources together to a multivariate time series describing the system of the DTS values, the electric current and the measured sea temperature. This leads

to an asynchronous time series with most of the variables unevenly sampled. The electric current and sea temperature are later referred to as control variables.

2.4 Neural Network Architectures

The neural network architectures most relevant to this thesis are reviewed in this section.

2.4.1 Recurrent Neural Networks and LSTMs

Recurrent neural networks (RNNs) are extensions of feedforward neural networks (FNNs) that allow feedback connections. The feedback connections enables memory of past observations, and lead to deep model structures. They are often used for sequence modelling tasks. Strengths of RNNs are that they make the order of observations explicit, and allow applying the same weights to each time step. This makes them flexible to use, as they can be applied to sequences of varying lengths. Figure 2.8 shows how unfolding the graph of applying an RNN to multiple timesteps leads to a deep structure.

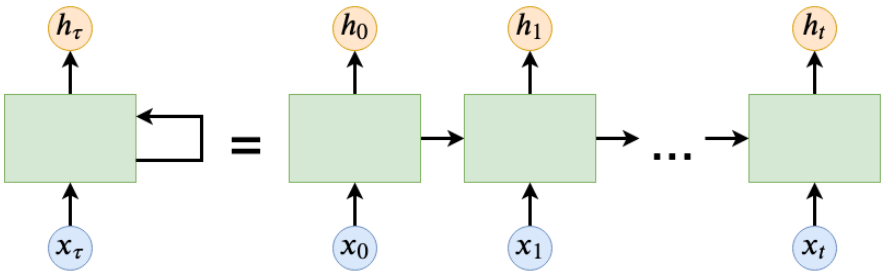


Figure 2.8: The deep structure of an unfolded RNN

A problem with RNNs is that they are susceptible to the problem of vanishing gradients, making it hard for the networks to learn long-term dependencies. To learn long-term dependencies, the error needs to backpropagate longer than for short term dependencies. The gradients will often decay on the way due to the repeated multiplication of the same weights. This makes for small updates, which gives slow training.

Long short-term memory networks

The introduction of long short-term memory networks (LSTMs) [Hochreiter and Schmidhuber, 1997] is responsible for much of the success of RNNs. A more

complicated recurrent cell with gating mechanisms allows propagating changes through many time steps, since gradients to a lesser degree vanish. LSTMs have a hidden state and a cell state that are transferred between time steps, and updates to these are controlled by the forget, input and output gates. The cell state can be thought of as the long running memory of the cell, while the hidden state is used for generating outputs and controlling the gates that control the state updates of the next unit.

The activation functions hyperbolic tangent and sigmoid are used in the LSTM cell, and can be seen in figure 2.9. The sigmoid outputs values between 0 and 1 and is used by the gating mechanisms to generate vectors that are applied through element-wise multiplication. An output value of 0 for a position means that no information is let through, while a value of 1 lets information though unchanged. Values in between decay the information to different degrees. The hyperbolic tangent outputs values between -1 and 1, regulating the values of the network.

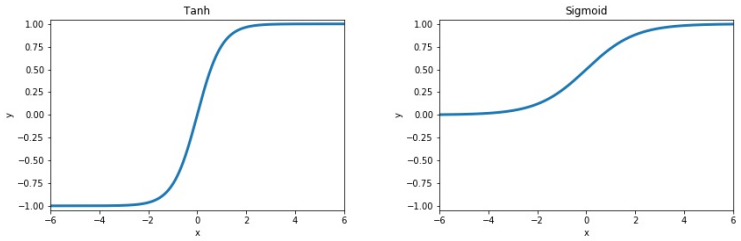


Figure 2.9: Hyperbolic tangent and sigmoid activation functions

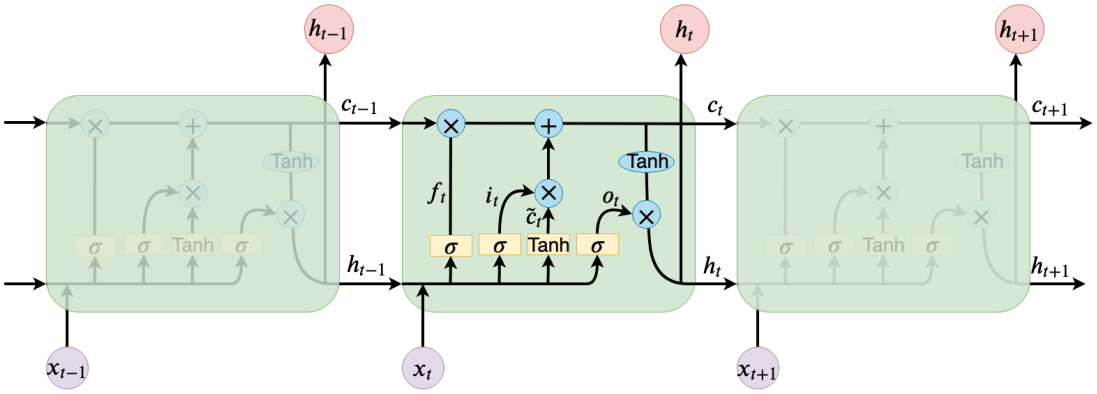


Figure 2.10: Unfolded LSTM. The cell state and hidden state are transferred from the previous unit. Gates control updates to the states based on the current input and previous hidden state. The new hidden state is used to generate outputs. The figure is based on a figure from [colah.github.io, 2019].

Figure 2.10 shows an unfolded LSTM with forget gate $\boldsymbol{f}$, input gate $\boldsymbol{i}$ and output gate $\boldsymbol{o}$. The gates have their own weights $\boldsymbol{W}_{xf}$, $\boldsymbol{W}_{hf}$, $\boldsymbol{W}_{xi}$, $\boldsymbol{W}_{hi}$, $\boldsymbol{W}_{xo}$ and $\boldsymbol{W}_{ho}$ and biases $\boldsymbol{b}_f$, $\boldsymbol{b}_i$ and $\boldsymbol{b}_o$. In addition the weights $\boldsymbol{W}_{xc}$ and $\boldsymbol{W}_{hc}$ and bias $\boldsymbol{b}_c$ controls the proposed addition to the cell state. The same weights and biases are applied to all time steps.

The input to an LSTM unit at a time step is the previous cell state $\boldsymbol{c}_{t-1}$, the previous hidden state $\boldsymbol{h}_{t-1}$ and the current sequence input $\boldsymbol{x}_t$. The sequence input and previous hidden state are used to calculate the gating vectors of the forget gate $\boldsymbol{f}_t$, input gate $\boldsymbol{i}_t$, and output gate $\boldsymbol{o}_t$. In the equations below σ denotes the sigmoid activation function.

$$\boldsymbol{f_t} = \sigma(\boldsymbol{x}_t\boldsymbol{W}_{xf} + \boldsymbol{h}_{t-1}\boldsymbol{W}_{hf} + \boldsymbol{b}_f) \tag{2.1}$$

$$\boldsymbol{i}_t = \sigma(\boldsymbol{x}_t\boldsymbol{W}_{xi} + \boldsymbol{h}_{t-1}\boldsymbol{W}_{hi} + \boldsymbol{b}_i) \tag{2.2}$$

$$\boldsymbol{o}_t = \sigma(\boldsymbol{x}_t\boldsymbol{W}_{xo} + \boldsymbol{h}_{t-1}\boldsymbol{W}_{ho} + \boldsymbol{b}_o) \tag{2.3}$$

A hyperbolic tangent activation calculates the proposed addition to cell state:

$$\tilde{\boldsymbol{c}}_t = tanh(\boldsymbol{x}_t\boldsymbol{W}_{xc} + \boldsymbol{h}_{t-1}\boldsymbol{W}_{hc} + \boldsymbol{b}_c) \tag{2.4}$$

The forget and input gates control the update of the cell state. The forget gate controls how much to keep of the previous cell state, while the input gate controls how much is added of the proposed addition $\tilde{\boldsymbol{c}}_t$. The symbol $\odot$ denotes element-wise multiplication.

$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t \tag{2.5}$$

The output gate controls the updated hidden state:

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot tanh(\boldsymbol{c}_t) \tag{2.6}$$

$\boldsymbol{h}_t$ is returned as the output at time t, and also forwarded to the next time step.

Many extensions of the LSTM have been proposed. Some extensions that facilitate asynchronous inputs are discussed in chapter 3.

2.4.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) are FNNs that depend on local connections and weight sharing to find location-invariant features. They typically consist of convolutional layers, activation functions and max pooling layers. Special for the

convolutional layers is that each neuron only is connected to a small region of the preceding layer, and that weights are shared between the neurons of a layer. This facilitates finding location-invariant features, and allows different input sizes.

The inputs to 2D convolutional networks have three dimensions; height, width and depth. In colored images, this corresponds to the height and width of the image, and the three color channels red, green and blue.

$$\begin{bmatrix} 1 & -1 & 1 \\ 1 & 1 & 1 \\ 1 & -1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 4 & 6 \\ 3 & 1 & 5 & 3 \\ 2 & 2 & 1 & 4 \\ 5 & 3 & 2 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 13 & 18 \\ 16 & 8 \end{bmatrix}$$

Figure 2.11: A 3×3 convolutional filter applied to a single channel input with stride 1. The $*$ symbolizes applying the convolutional operation with the left matrix as the filter to the right matrix.

Convolutional layers consist of filters that are applied to small regions of the input. Each filter consists of weights and a bias. The filter is slid over smaller regions of the input, across the height and width. It calculates the dot product of its weights and the inputs across all input channels. The filter has the same depth as the input, and a width and height that can be decided. The number of filters in a layer decides the depth of the representation produced by the layer. The stride decides how far the filter slides before doing a new calculation. The edges of the input can be padded with zeroes to conserve the size of the representation through the convolutional operation. Figure 2.11 shows the result of applying a convolutional filter with weights and 0 bias to an input matrix with depth 1, using a stride of 1.

$$\begin{bmatrix} 1 & 2 & 4 & 6 \\ 3 & 1 & 5 & 3 \\ 2 & 2 & 1 & 4 \\ 5 & 3 & 2 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 3 & 6 \\ 5 & 4 \end{bmatrix}$$

Figure 2.12: 2×2 max pooling with stride 2

A max pooling layer reduces the spatial dimension of its input. The max pooling layer has a size and stride, and outputs the maximum value of its input field. Figure 2.12 shows how max pooling with a filter size of 2×2 reduces the spatial dimension of the input.

All the operations of the convolutional and max pooling layers are differentiable, so the usual optimization strategies based on backpropagation of a loss function can be used.

The CNNs used for image processing use 2D convolutional layers that convolve over the height and width of images. 1D convolutional layers can be used for inputs with a single spatial or temporal dimension, and have been successfully applied to language modelling, for example in Collobert and Weston [2008]. A 1D convolutional layer is a 2D convolutional layer with a height of 1.

2.4.3 Autoencoders

Autoencoders are neural networks trained to reproduce their inputs. They consists of encoder and decoder parts. The encoder constrains the network to produce a more sparse representation of the input, and the decoder tries to reconstruct the input from the representation. For some applications, the representation of reduced dimensionality is what is of interest from the autoencoder. In other applications, the reconstruction error is interesting. This will be shown can be the case for anomaly detection.

An undercomplete autoencoder is shown in figure 2.13. Undercomplete autoencoders are FNNs with fewer units in the hidden layers than in the input layer, and thus must learn how to efficiently represent the inputs in order to optimally reproduce them.

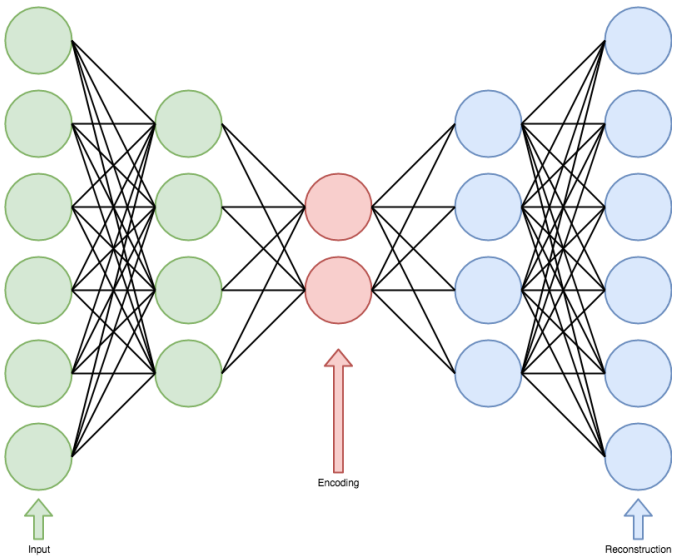


Figure 2.13: Undercomplete autoencoder

LSTM autoencoders

LSTMs can also be used in autoencoders. Sutskever et al. [2014] introduced using LSTMs for sequence to sequence learning, and showed how they could be used for machine translation. In sequence to sequence learning, one LSTM is used to encode an input sequence to a vector representation of fixed length. Another LSTM decodes the target sequence from the input. For an LSTM autoencoder, the target sequence is equal to the reversed input sequence. Reversing the sequence makes learning easier, as the model can begin with looking at short range correlations [Srivastava et al., 2015].

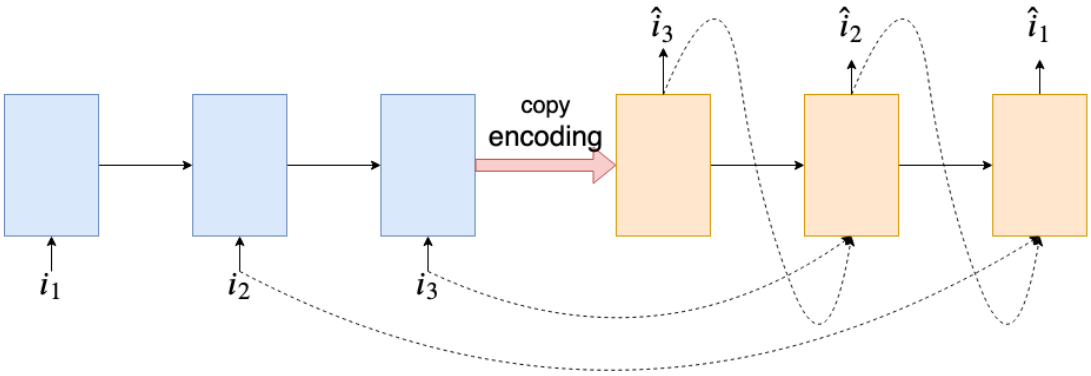


Figure 2.14: LSTM autoencoder. The blue boxes represent the encoder LSTM cell and the orange boxes the decoder LSTM cell.

The decoder can be either conditional or unconditional. A conditional decoder receives the last generated output as input, while an unconditional decoder does not receive any input. Figure 2.14 shows an LSTM autoencoder. The conditional version of the decoder uses the i_js as input during training and $\hat{i}_j$s as inputs during inference, as shown with dotted arrows. The unconditional decoder only uses the states to produce output. The figure also shows how the input sequence is reconstructed in reversed order.

Convolutional autoencoders

Also autoencoders based on CNNs have been proposed [Masci et al., 2011]. Here the advantages of CNNs are applied to feature learning in an autoencoder architecture. Local connections and shared weights lead to preserved spatial locality.

The encoder part of a convolutional autoencoder (CAE) typically consists of convolutional layers followed by max pooling layers, this way producing an encoding of reduced dimensionality. Equation 2.7 shows how the representation of the k'th channel $\boldsymbol{H}_k$ is produced by the 2D convolutional operation with filter k. $\boldsymbol{X}$

is the input, σ an activation function, W_k the weights of the k'th filter and b_k the bias that is broadcast to all the values. The symbol $*$ denotes 2D convolutions.

$$\boldsymbol{H}_k = \sigma(\boldsymbol{X} * \boldsymbol{W}_k + b_k) \tag{2.7}$$

The decoder layer tries to reverse the operations of the convolutional layer to reconstruct the input. For this purpose transposed convolutional layers followed by upsampling layers can be used. The transposed convolutional layer does not perform the mathematical inverse of the convolutional operation, but rather performs the normal convolutional operation with extra padding to increase the size instead of decreasing it. Masci et al. [2011] use the same weights $\tilde{W}$ in the decoder as in the encoder, but they are flipped over both dimensions. Equation 2.8 shows the transpose convolution operation. Again $*$ symbolizes the 2D convolutional operation. Depending of the filter sizes and output shape, it might add padding. The reconstruction $\tilde{\boldsymbol{X}}$ is obtained by applying 2D convolutions to all the channels and summing the result, adding a bias and feeding the result through an activation function.

$$\tilde{\boldsymbol{X}} = \sigma(\sum_{k \in H} \boldsymbol{H}_k * \tilde{\boldsymbol{W}}_k + c) \tag{2.8}$$

Upsamling reverses the dimensionality reduction preformed by the max pooling layers. An example of an upsampling operation is to broadcast values, and an example can be seen in figure 2.15. Here each value is broadcast to 2×2 values, and such the spatial dimension of original input is regained after 2×2 max pooling with stride 2. All the values are not regained, which is not possible to perfectly achieve from the smaller representation.

$$\begin{bmatrix} 1 & 2 & 4 & 6 \\ 3 & 1 & 5 & 3 \\ 2 & 2 & 1 & 4 \\ 5 & 3 & 2 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 3 & 6 \\ 5 & 4 \end{bmatrix} \longrightarrow \begin{bmatrix} 3 & 3 & 6 & 6 \\ 3 & 3 & 6 & 6 \\ 5 & 5 & 4 & 4 \\ 5 & 5 & 4 & 4 \end{bmatrix}$$

Figure 2.15: 2×2 max pooling with stride 2, followed by the upsampling by broadcasting to regain the input size.

2.5 Principal Component Analysis (PCA)

Principal component analysis is a powerful linear dimensionality reduction technique based on the singular value decomposition (SVD) from linear algebra. The

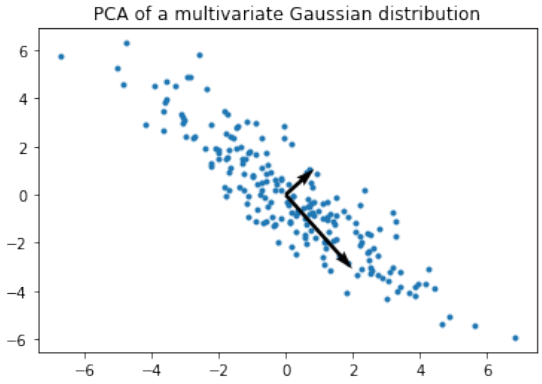


Figure 2.16: PCA applied to points generated by a multivariate Gaussian distribution. The arrows show the right singular vectors, with the longest arrow corresponding to the first principal component.

SVD of a $n \times p$ matrix $\boldsymbol{X}$ of rank r can be seen in equation 2.9. $\boldsymbol{X}$ has n observations of p features.

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \sigma_1\boldsymbol{u}_1\boldsymbol{v}_1^T + \sigma_2\boldsymbol{u}_2\boldsymbol{v}_2^T + ... + \sigma_r\boldsymbol{u}_r\boldsymbol{v}_r^T \tag{2.9}$$

$\boldsymbol{\Sigma}$ is an $n \times r$ rectangular diagonal matrix with the singular values σ_i on the diagonal. $\boldsymbol{U}$ is $n \times n$ and has orthonormal columns $\boldsymbol{u}_i$ called the left singular vectors. $\boldsymbol{V}$ is $p \times r$ and has orthonormal columns $\boldsymbol{v}_i$ called the right singular vectors.

What makes the SVD useful for dimensionality reduction is that the singular values can be ordered by importance, such that $\sigma_1 > \sigma_2 > ... > \sigma_r$. Each direction $\boldsymbol{v}_j$ of the SVD accounts for $\sigma_j^2 / \sum_{i=1}^{r} \sigma_i^2$ of the variance in the data. Using the k first components $\boldsymbol{X}_k$ as shown in equation 2.10 gives the best rank-k approximation to $\boldsymbol{X}$ [Strang, 2016].

$$\boldsymbol{X}_k = \boldsymbol{U}_k\boldsymbol{\Sigma}_k\boldsymbol{V}_k^T = \sigma_1\boldsymbol{u}_1\boldsymbol{v}_1^T + \sigma_2\boldsymbol{u}_2\boldsymbol{v}_2^T + ... + \sigma_k\boldsymbol{u}_k\boldsymbol{v}_k^T \tag{2.10}$$

From a statistical point of view, the first component has the maximal variance [Rencher and Christensen, 2012]. It finds the dimension where the observations are maximally spread out. The succeeding principal components have the maximal possible variance, given that they are orthogonal to the all the preceding components. Figure 2.16 shows how PCA finds orthogonal dimensions with maximal variance.

PCA leads to the transformation $\boldsymbol{t}_k = \boldsymbol{V}_k^T\boldsymbol{x}$ that uses the top k right singular vectors from the SVD. The transformation leads to an encoding $\boldsymbol{t}_k = (t_1, t_2, ..., t_k)$ of length k. To reconstruct the data from the encoding, the inverse transform $\boldsymbol{x}_k = \boldsymbol{V}_k\boldsymbol{t}_k$ is applied. The error in the reconstruction is $\boldsymbol{e} = \boldsymbol{x}_k - \boldsymbol{x}$.

2.6 Linear Regression

Linear regression models a linear relationship between a dependent variable y and a number p of explanatory variables $\boldsymbol{x}$, such that:

$$y_i = w_0 + w_1 x_{i,1} + w_2 x_{i,2} + ... + w_p x_{i,p} + \epsilon_i$$

The dependent variable is assumed to be a linear combination of the explanatory variables with an added error term. The error term can be due to measurement errors, and the errors are assumed to be of constant variance.

A prediction $\hat{y}_i$ of y_i is given by

$$\hat{y_i} = w_0 + w_1 x_{i,1} + w_2 x_{i,2} + ... + w_p x_{i,p} \tag{2.11}$$

The weights w_j are fit to minimize $(\boldsymbol{y} - \hat{\boldsymbol{y}})^2$ for the training data. This can be solved analytically, and the best weights are given by equation 2.12 [Russell and Norvig, 2016]. $\boldsymbol{X}$ is an $n \times p$ matrix of the explanatory variables of the training data, and $\boldsymbol{y}$ is a vector of length n with the corresponding dependent values.

$$\boldsymbol{w} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y} \tag{2.12}$$

2.7 Training and Evaluation Scheme

This section introduces how machine learning models with time series can be trained and evaluated. First the sliding window approach to constructing multiple samples from a long sequence is presented, and then variations of cross validation for time series are discussed.

2.7.1 The Sliding Window Approach

The sliding window approach is a method used in time series analysis for constructing multiple samples from a long time series. Samples are constructed by sliding a fixed-sized window along the time series. When the window slides over a sub-sequence it is used to construct samples. The sub-sequences might be overlapping depending on the stride used when constructing samples. The stride describes how many time steps to slide between each sample, and often a stride of one is used to maximize the number of samples. For prediction problems, each window is further subdivided into inputs and targets, both of a fixed size. Thus the samples contain a fixed number of observations as inputs and a fixed number of the next consecutive observations as targets. Figure 2.17 shows how the sliding window approach can be used to construct samples.

Original sequence: abcdefgh

1) abc, bcd, cde, def, efg, fgh
2) abc, cde, efg
3) ab $\rightarrow$ c, bc $\rightarrow$ d, cd $\rightarrow$ e, de $\rightarrow$ f, ef $\rightarrow$ g, fg $\rightarrow$ h

Figure 2.17: The sliding window approach used to constructing multiple samples from the sequence *abcdefgh*. 1) Uses a stride of 1 to construct overlapping samples of length 3. 2) Uses a stride of 2 to construct overlapping samples of length 3. 3) Uses a stride of 1 to construct overlapping samples for prediction with inputs of length 2 and targets of length 1.

For time series analysis, an alternative to using the sliding window approach is to consider the whole time series as it is. Advantages of using the sliding window approach instead, is that it allows for many of the techniques that speed up training of deep learning models and also makes them more successful. In particular, constructing individual samples allows for training of neural networks in batches. It also allows randomization of the samples in each epoch, and the individual samples facilitate the cross-validation techniques described in the next section.

Disadvantages of the sliding window approach include that it constrains how much previous history is available to the models when making predictions. Thus an assumption needs to be made of how many timesteps of previous history is sufficient for the task at hand. Also the constructed samples will not be independent if overlapping sub-sequences are used during construction. Care must be taken when training and evaluating, such that the same data is not used both for training and testing.

2.7.2 Cross Validation

One of the biggest challenges in machine learning is producing models that not only perform well on the data seen during training, but also generalize well to unseen data. This is usually evaluated during model construction by dividing the available data into three distinct sets called the training, validation and test sets. The training data is used for fitting the model parameters by back-propagating the error on the training set to update the network weights. A validation data set is often used to keep from overfitting the model to the training data. The validation performance can be used for early stopping, meaning that the weight configuration best performing on the validation data is kept, even though the weights can be

tweaked to fit even better to the training data. Performance continuing to increase for the training data and decreasing for the validation data is a clear sign of overfitting.

Even though the validation data is not used directly to update the weights, it does influence the model selection. It will give a biased measure of the models performance on unseen data, as information about the validation set leaks into the training procedure. Thus a separate test set is usually set aside and used only for final evaluation of the trained model.

Traditionally, the last consecutive block of data was set aside for testing when constructing time series prediction models [Bergmeir and Benitez, 2012]. A disadvantage of splitting the data into training, validation and test data is that not all the available data is used for fitting the model. Also not all the data is used for evaluation, and a small test set might not be representative of the true data distribution. A procedure called k-fold cross validation tries to mitigate these issues. The data is randomly split into k parts of approximately equal size. In k rounds, one of the k parts is kept for evaluation while the other k-1 parts are used for training the model. Each part is used for evaluation in one of the rounds, and the average performance is reported. This way all of the data is used both for training and evaluation which gives a more robust measure of the performance.

An extension to this is cross validation with both validation and test sets. In this approach, one of the k parts is used for testing, one part is used for validation and the other $k - 2$ parts are used for training. As previously explained, using a test set that has not influenced the training in any way gives a less biased estimate of the performance on unseen data.

For time series analysis of one single time series, this picture becomes somewhat more complicated than for problems with independent samples. The reason for this is that the samples are more interconnected. The sliding window approach is often used to construct overlapping samples. This leads to consecutive samples not being independent, as many of the same time steps occur in multiple samples. Awareness of this is crucial during training, as the samples in the training, validation and test sets should be independent.

Inspired by the findings of Bergmeir and Benitez [2012], blocked 5-fold cross validation with validation and test sets is employed for training and evaluation in the experiments. Blocked cross validation differs from regular cross validation in that the k parts are made up of contiguous data, as shown in figure 2.18. This guarantees that each time step only contributes to either training, validation or testing in each fold. According to Bergmeir and Benitez [2012], using cross validation leads to more robust validation and model selection compared to using only the last contiguous block for testing. They find no practical effect of theoretical problems that could invalidate using data from the past to test models trained

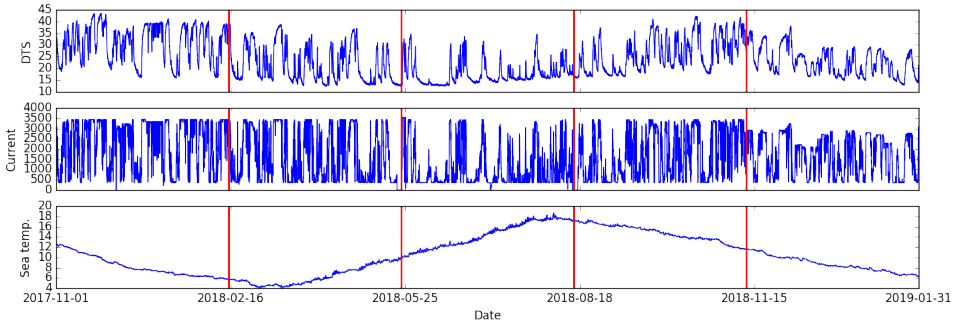


Figure 2.18: Division into folds by 5-fold blocked cross validation in the export cables.

with future data.

2.8 Evaluation Metrics and Statistical Significance

This section reviews the evaluation metrics used in the experiments, and the statistical test used to test the significance of the experimental results.

2.8.1 Evaluation Metrics for Time Series Prediction

The mean squared prediction error is used to evaluate the prediction models of chapter 4. Squaring the error heavily penalizes big errors compared to multiple, smaller errors. The measure is also used as the loss function in the trained neural networks.

2.8.2 Evaluation Metrics for Anomaly Detection

The problem of anomaly detection is an instant of binary classification with the classes ”anomaly” and ”normal”. Anomaly detection problems with real world data often have imbalanced class distribution. Most of the data is normal and constitutes negative samples. A few anomalies may be present, and these constitute positive samples. The task of the anomaly detection model is to return as many of the positive samples as possible, while limiting the amount of negative samples returned.

Positive samples that are correctly classified are called true positives, while negative samples that are wrongly classified as positive are false positives. Similarly a positive sample that is wrongly classified as negative is a false negative, and

a negative sample correctly classified is a true negative. These definitions are summarized in table 2.1.

	positive sample	negative sample
classified as positive	true positive	false positive
classified as negative	false negative	true negative

Table 2.1: Definition of anomaly detection outcomes

Two types of errors can be made by an anomaly detection model. The first type is false negatives, which corresponds to not alerting of an anomaly when there is one. The other type is false positives, which corresponds to falsely alerting of an non-existing fault.

For many classification models, a threshold value has to be set to decide the cutoff for classifying samples as negative or positive based on the model output. By varying this threshold, the trade-off of having few false positives or few false negatives can be somewhat controlled, as the rates of false positives and false negatives can change based on the threshold setting. A way to visualize this is to use a receiver operating characteristic curve (ROC-curve). The ROC-curve plots the true positive rate against the false positive rate for different threshold values. True positive rate is defined in equation 2.13 and false positive rate is defined in equation 2.14.

$$\text{true positive rate} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \tag{2.13}$$

$$\text{false positive rate} = \frac{\text{false positives}}{\text{false positives} + \text{true negatives}} \tag{2.14}$$

The measure AUC (area under the curve) is the area under the ROC-curve, and is an aggregated measure of model skill across threshold values. The maximum possible AUC is 1 for a perfect model. A skill-less model will produce an ROC-curve with a diagonal line, and will have an AUC of 0.5.

AUC is used to evaluate the anomaly detection models of chapter 5. It is chosen because a balanced set of equally many anomalous and normal samples is used for evaluation. Had the dataset been highly imbalanced with fewer anomalous samples, it would have been more appropriate to use a precision-recall curve, and calculate the area under that curve instead. Precision is defined in equation 2.15, while recall is equivalent to true positive rate. Precision does not use the number of true negatives in its calculation, which makes is more appropriate than false positive rate for imbalanced data sets.

$$\text{Precision} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}} \tag{2.15}$$

2.8.3 Establishing Statistical Significance

In the experiments of chapters 4 and 5, statistical hypothesis tests are executed to test whether the seemingly best model is better than the other tested models with statistical significance. The Wilcoxon signed-rank test is used [Wilcoxon, 1945].

The Wilcoxon signed-rank test is a hypothesis test that assesses whether two paired samples stem from distributions with different population means. The null hypothesis is that the differences of the pairs are normally distributed around zero. For a two-sided test, the alternative hypothesis is that they do not center around zero. If the p-value of the test is lower than the chosen significance level, the null hypothesis is rejected. It is then believed that the differences are not centered around zero, and one population has to have a larger mean than the other. If the p-value is not lower than the chosen significance level, the null hypothesis cannot be rejected. That does not mean than the null hypothesis is proven to be true, only that it cannot be disproven.

The test entails ranking the absolute values of the paired differences. The ranks of the negative differences receive a negative sign, and all the ranks are summed. The test statistic is the sum of the signed ranks. The test statistic is compared to a critical value from a statistical table to decide whether the null hypothesis should be rejected. Alternatively, a p-value can be calculated. The statistical software R [R Core Team, 2017] and the function `wilcox.test` is used to calculate p-values in the tests executed in this thesis. The tests use a significance level of 0.05.

The Wilcoxon signed-rank test is chosen as it assesses paired samples, and the experiments of both chapters 4 and 5 yield results that can be assessed as paired samples. Each model will be evaluated multiple times under certain constraints. Evaluations of two models under the same constraints make up a paired sample, which is further explained where the test is applied. The test is chosen as it is the most appropriate test found. It does not assume that the variables themselves are normally distributed. It takes the magnitude of the samples into account, not just whether one is larger than the other. It does however assume that the pairs are independent. The paired results from the experiment will not be completely independent, which also is discussed where the test is applied.

Chapter 3

Related work and Motivation

This chapter presents previous research related to the thesis. A structured literature review was performed, and is detailed in section 3.1. The related works that were uncovered are presented in section 3.2. Section 3.3 summarizes the related work and connects it to the export cables, while section 3.3 summarizes the motivating elements of the thesis.

3.1 Structured Literature Review

A structured search inspired by Kofod-Petersen [2015] was performed as a formal and systematic way of exploring research relevant towards finding answers to the research questions stated on page 2. The research questions are repeated here for completeness:

Research question 1 *How are the asynchronous time series of the export cable system best handled in sequential deep learning models?*

Research question 2 *Does exploiting relationships in both the temporal and spatial dimensions of the export cable data give more robust anomaly detection than only regarding one of the dimensions?*

Some advantages of reviewing literature in a structured way are that it helps avoiding biased work, it helps identifying gaps of knowledge and it can highlight areas where additional research is needed [Kofod-Petersen, 2015].

To work towards answering the research questions, related literature was read to cover the topics of asynchronous time series in deep learning and deep learning applied to anomaly detection in time series. The following set of literature review questions were designed to uncover related work relevant to the research questions, and guide the literature review:

Q1 Has deep learning based anomaly detection been applied successfully to DTS systems?

Q2 What approaches exist to unsupervised anomaly detection in time series using deep learning?

Q3 What other deep learning approaches to anomaly detection in time series exist with other degrees of supervision, and can they be transformed to be used in an unsupervised setting?

Q4 How have asynchronous time series been handled in sequential deep learning models?

The review protocol of the following section describes the steps taken to find the related work presented in the later sections of this chapter, with the aim of making the literature review procedure reproducible.

3.1.1 Protocol

Google Scholar was used as a search engine to find related literature. This decision was made as Google Scholar searches across many different publication databases and other web pages. It tries to rank the results the way a researcher would do, weighing where the text was published, who it was written by, its citation history and the full text of the documents [scholar.google.com, 2019]. The searches were done without including patents and citations, as they were not of interest. The results were returned sorted by relevance by the search engine. No restriction was put on the publication year of the articles, as the search engine ranks partly based on how often and how recently articles have been cited in other scholarly literature, and this was judged to be a sufficient measure to ensure finding recent relevant work. Only full research papers written in English were considered, excluding isolated abstracts and review articles. Duplicate papers were ignored. The 20 first hits of each search were reviewed, as thousands of papers were returned and it was not plausible to review all. Separate searches were executed to find literature covering each of the literature review questions. The search terms employed for each question are presented in table 3.1.

When 20 articles were obtained from each search, an inclusion screening was done to trim away work not relevant to the thesis. Different inclusion criteria were used to screen the results for the different searches. Some articles could be included or discarded by only reviewing the abstract, while for some papers the whole text had to be surveyed to see if it met the relevant inclusion criterion. Works that were judged not to meet a criterion were excluded from further consideration. The

Question	Search terms
Q1:	(anomaly OR outlier OR fault) AND ("deep learning" OR LSTM OR RNN OR GRU OR CNN OR "machine learning") AND "distributed temperature sensing"
Q2:	(anomaly OR outlier OR fault) AND ("deep learning" OR LSTM OR RNN OR GRU OR CNN OR "machine learning") AND "time series" AND unsupervised
Q3:	(anomaly OR outlier OR fault) AND ("deep learning" OR LSTM OR RNN OR GRU OR CNN OR "machine learning") AND "time series"
Q4:	("heterogeneous time series" OR "asynchronous time series" OR "unevenly spaced time series") AND ("deep learning" OR LSTM OR RNN OR GRU OR CNN OR "machine learning")

Table 3.1: Search terms used to find related work as part of the structured literature review.

Question	Inclusion criteria
Q1:	Concerns deep learning applied to anomaly detection in a DTS system
Q2:	Concerns deep learning applied to unsupervised anomaly detection in time series
Q3:	Concerns deep learning applied to anomaly detection in time series
Q4:	Concerns applying deep learning techniques to asynchronous time series

Table 3.2: Inclusion criteria employed in the structured literature review.

criteria were set to exclude work that did not contribute towards answering the literature review questions and guide towards the research goal.

For the fourth literature review question concerning asynchronous time series, some additional articles are included in the review of related work in addition to the articles found by the structured search. These articles did not appear through the search, but were referenced as related work in some of the reviewed articles, and proved to be very relevant upon inspection. In particular, this applies to the reviewed works of Neil et al. [2016] and Baytas et al. [2017].

3.2 Review of Related Work

This section presents the related work uncovered in the literature review. Section 3.2.1 presents the results of the search related to Q1 which concerns deep learning based anomaly detection applied to DTS systems. Section 3.2.2 presents works regarding deep learning used for anomaly detection in other domains, and contains the works resulting from the searches connected to questions Q2 and Q3. Previous research on asynchronous time series in deep learning is presented in Section 3.2.3 and is the result of the search of Q4.

3.2.1 Anomaly Detection Applied to DTS Systems

Only a single paper came through the search on applying deep learning to anomaly detection in DTS systems. It only contains "initial proof-of-concept results", and very little details on the anomaly detection approach or machine learning models tested.

Araujo et al. [2018] approach detecting leaks from liquid pipelines by applying machine learning to the signals of the DTS system monitoring the pipelines. With 30km of pipelines and DTS samples spaced 0.5 meters in space and 8 minutes in time, they are facing a problem with very similar characteristics to the one of this thesis. They compare machine learning approaches to the traditional approach of setting value thresholds to detect anomalous values. They argue that machine learning is a more suited approach than thresholding. This is due to serviceable thresholds being difficult to set, and might need to be changed in answer to changes in the surroundings. They also remark of the advantage of using DTS systems for monitoring the pipelines. Compared to other solutions, analysis of DTS system data offers analysis of higher resolution. This allows better localization of leaks.

They train and compare a number of models. The different models are trained using data exhibiting only normal behaviour. The datasets for both validation and testing contain some leak events. They explore both statistical and machine learning methods, and find that singular value decomposition (SVD) is unsuited to

detect the subtle changes caused by the pipeline leaks. It was reportedly computationally expensive and better suited to small sections of pipeline without much variation. They further report investigating several different machine learning techniques that include LSTMs, CNNs, autoencoder architectures and shallow ML-classification approaches. They do not report any further specifics of the models nor how the different models compared, only that they decided to go with a convolutional undercomplete autoencoder model that incorporates both spatial and temporal information. It is not clear how this model is used to detect anomalies.

3.2.2 Anomaly Detection

More work was found on deep learning for anomaly detection in general, not only applied to DTS data. Few papers explicitly treat unsupervised anomaly detection.

Malhotra et al. [2015] propose a semi-supervised method for anomaly detection in time series. It uses deviation from predicted normal behaviour to detect anomalies. The prediction model consists of stacked LSTMs. Two LSTM cells are stacked by using the output of the first LSTM as the input to the one stacked above it. The cell state is not passed between the two LSTM cells. Using stacked LSTM cells may allow the network to better capture the features of the time series, similar to how deep FNN architectures benefit from hierarchically building abstract representations of features with increasing network depth.

The model is trained with non-anomalous data to model the normal behavior. The utility of this approach is motivated by real-word applications where plenty of non-anomalous data often is available. Data with anomalous behaviour may however be rare, as in the export cables of this thesis. The modelled normal behaviour is used to make predictions for a number of time steps. The prediction errors from a validation set are modelled by a multivariate Gaussian distribution, which is used to calculate the likelihood of anomalous behavior. Validation sets of both normal and anomalous data are used to set a threshold to judge whether observations are normal or anomalous based on the anomaly likelihood.

In Malhotra et al. [2016], the idea of using LSTMs for anomaly detection has been further matured. They propose an LSTM autoencoder that is trained to reconstruct time series exhibiting normal behaviour. The reconstruction error obtained from comparing the original time series to its reconstructed version is used to detect anomalies.

They report that this model better allows normal phase changes due to external factors that are not well handled by the LSTM prediction approach. Time series may be inherently unpredictable due to manual controls or un-monitored environmental controls. In these situations, the prediction-based anomaly detection model might not be suitable. When the phase changes are unpredictable, they can lead to unpredictable behavior that is well within what is normal.

The model consists of an encoder that learns to produce a fixed dimension representation of the input time series, and a decoder that learns to reconstruct the input from the encoding produced by the encoder. The encoding produced is the final state of the encoder, which is used as the initial state of the decoder. A linear layer transforms the output of the LSTM decoder to predict the target. The encoder and decoder are jointly trained using only non-anomalous time series. The reconstruction error of a sequence is used to compute an anomaly likelihood score. The score is a proxy of the likelihood of the sequence being anomalous. The intuition is that the model has only seen normal sequences during training, and thus is better at reconstructing normal sequences. In a supervised setting, an anomaly threshold is set based on both normal and anomalous data.

They apply their model to four datasets, where the application to an ECG dataset is of the biggest relevance to this thesis. In this application they do not have anomalous data for setting the threshold, instead showing that the model can be used in an unsupervised setting. Here they set the threshold based on statistics of the anomaly likelihood scores from a validation set of only normal data.

They compare the LSTM autoencoder model to their previously proposed LSTM prediction model, and find that for predictable datasets the prediction based model is superior. For datasets that are inherently unpredictable due to external factors, the LSTM autoencoder model does considerably better.

An application of the prediction-based LSTM anomaly detection model proposed by Malhotra et al. [2015] is seen in Filonov et al. [2016], where they apply the model to detection of cyber-attacks in a gasoil plant. They focus on multivariate industrial time series that contain both data from sensors and control signals. As they do not have such a dataset available, they create a mathematical model of a real gasoil plant. They use this to generate data of normal behaviour, but they also modify some of the process logic to simulate a cyber attack, and this way generate data containing anomalies.

Instead of fitting a distribution to the prediction errors as done by Malhotra et al. [2015], they use the prediction error directly. A sequence is marked as anomalous if the prediction error exceeds a threshold τ. They first report setting τ to be the 0.999-quantile of the empirical error distribution. Then they vary the threshold and compare the precision as recall at different thresholds. They consider the threshold to be a tunable parameter that can be used to control the rate of false negatives versus the rate of false positives.

Chauhan and Vig [2015] present a direct application of Malhotra et al. [2015] to electrocardiography (ECG) signals with the goal of detecting multiple arrhythmias. They claim that advantages of the LSTM prediction approach compared to previous techniques used to analyse ECG signals, are that minimal pre-processing of the data is needed. It does not require hand coded features nor prior knowledge

of the anomalies to work.

Yan and Yu [2015] address anomaly detection in gas turbine combustors. They learn features from sensor measurements using stacked denoising autoencoders[1] (SDEA). Anomaly detection using the extracted features is compared to anomaly detection using features that were handcrafted based on engineering and domain knowledge. They want to avoid handcrafting features, as the crafted features will be very problem specific, and the approach does not scale. Anomaly detection is done by feeding the features to an extreme learning machine classifier[2] that outputs the probability of an anomaly. The data is highly imbalanced with a lot more samples of normal behavior than anomalous behaviour. To overcome the class imbalance, the anomalous samples are more heavily weighted during training.

Liu et al. [2017] apply deep learning to anomaly detection in rotating electric machines. They want to detect faults to reduce maintenance costs and prevent accidents. Previous methods depend on handcrafted feature extraction, which heavily depends on prior knowledge and human labor. They claim that industry structures and data are becoming increasingly complicated, making handcrafting features an even less attractive option. They are considering rotation machinery, which results in strongly periodic data. They propose a CNN-structure that takes as input a matrix of shifted subsets of the input time series. To allow direct comparison of the same phases of the period, the length between the shifted time series corresponds to a multiple of the period of the signal.

3.2.3 Asynchronous Time Series Data

Finally the related works on asynchronous time series data in deep learning are presented. Zhang et al. [2017] address rare event prediction with multivariate, non-uniformly sampled time series data consisting of both continuous and categorical variables. The nature of the data and complex dependencies between the variables make the prediction problem challenging. Previous approaches have relied on handcrafted features, or breaking up the problem into synchronous variables. Their proposed approach involves symbolizing the input and feeding the symbolized words to an LSTM. The representation of the vocabulary is trained jointly with the LSTM to learn discriminative features for the given task. Further they cast the prediction problem as a classification problem, with different classes corresponding to different times until the next event of interest.

Neil et al. [2016] approach the problem of traditional RNN models being ill-suited to process asynchronous and unevenly sampled time series data. Real world

[1]Denoising autoencoder are autoencoders where some of the inputs are randomly masked during training.

[2]Extreme learning machine (ELM) is a version of a two-layer FNN where the hidden nodes are randomly chosen while the output weights are analytically determined[Huang et al., 2016].

scenarios where multiple sensors with different sampling rates give rise to asynchronous time series motivate their model. In these scenarios, enforcing constant update rates negatively affect the precision and efficiency of RNNs, due to the effects of resampling.

To deal with these issues, they introduce the Phased LSTM (PLSTM). It is an extension to the LSTM that adds a time gate in addition to the forget, input and output gates. The time gate is controlled by independent rhythmic oscillations. In one of the phases of the oscillations the time gate is closed, which does not allow any updates to the states of the LSTM cell. In the other phases it is partly or fully open, allowing updates to the states. The oscillations are specified by three parameters that can be learned or set by the user. The parameters are the period of the oscillations τ, the phase shift s and r_{on}, which controls the ratio of the open phase to the duration of the period. They propose the following calculation of the time gate k_t:

$$\phi_t = \frac{(t-s)\mathrm{mod}\tau}{\tau}, \qquad k_t = \begin{cases} \dfrac{2\phi_t}{r_{on}} & \text{if } \phi_t < \dfrac{1}{2}r_{on} \\ 2 - \dfrac{2\phi_t}{r_{on}} & \text{if } \dfrac{1}{2}r_{on} < \phi_t < r_{on} \\ \alpha\phi_t & \text{otherwise} \end{cases} \tag{3.1}$$

The parameter α is the leak rate. It is used in the closed phase, and allows propagation of important gradient information when the time gate is closed. The time gate k_t controls updates to the cell state c_t and hidden state h_t as presented in equations 3.3 and 3.5. When the time gate is fully open ($k_t = 1$), the updates are identical to a normal LSTM. As the gate closes, less of the proposed updates are applied, and more of the old states are kept.

$$\tilde{\boldsymbol{c}}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t * tanh(\boldsymbol{W}_c \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_c) \tag{3.2}$$

$$\boldsymbol{c} = k_t \odot \tilde{\boldsymbol{c}}_t + (1 - k_t) \odot \boldsymbol{c}_{t-1} \tag{3.3}$$

$$\tilde{\boldsymbol{h}}_t = \boldsymbol{o}_t \odot tanh(\tilde{\boldsymbol{c}}_t) \tag{3.4}$$

$$\boldsymbol{h}_t = k_t \odot \tilde{\boldsymbol{h}}_t + (1 - k_t) \odot \boldsymbol{h}_{t-1} \tag{3.5}$$

They report favorable results of the PLSTM compared to an LSTM with timestamps of observations as an additional feature. It is reported being superior on signals sampled asynchronously and at high resolution.

An effort towards making LSTMs accept univariate time series with uneven sampling rates is presented by Baytas et al. [2017]. They are concerned with analyzing patient records with very irregular sampling intervals, raging from days to years between recorded events. They want to cluster patients into disease

characterizing subtypes, with the aim to offer the most suitable type of treatment. In healthcare, the time between two consecutive hospital visits is an important factor of the decision making, and thus they require a model that takes the elapsed time between records into account. They propose an LSTM modification that they call time-aware LSTM (T-LSTM) to meet their needs. It has the LSTM cell state decomposed into short-term and long-term parts. Based on the elapsed time, the effect of the short term part is discounted. The altered previous cell state is used to update the cell state and the hidden state.

They report that the T-LSTM is able to learn powerful representations of temporal patient journals that they use to cluster the population. They compare their approach to two LSTM extensions that alter the forget gate to take time into account, and shows that their approach learns more powerful features.

Binkowski et al. [2018] address prediction of multivariate asynchronous time series. They are concerned with financial data, where the same signal can be observed from different sources in asynchronous moments of time. These observations might be correlated, and the significance of each past observation upon the prediction can change over time. They are inspired by the relatively good performance of traditional econometric models such as autoregressive models[3]. Inspired by these they propose the significance-offset convolutional neural network, which they see as a neural network extension of standard autoregressive models. The model uses a nonlinear weighting mechanism, such that the final output prediction is obtained as a weighted sum of adjusted inputs. The weights are data-dependent functions learned through a CNN.

They further motivate their network structure by observing that many real-world applications give rise to multivariate time series, where the different dimensions are observed separately and asynchronously. The much used pre-processing step of aligning the dimensions at fixed frequencies may lead to loss of information if a frequency too low is used, or prohibitive enlargement of the data set. They rather propose decomposing all the dimensions to a single dimension, with dimension marker and duration as additional features. This representation is used in their proposed model.

They compare their model to LSTM, PLSTM and multilayer CNN on several datasets, and find that their model outperforms the others on all tested asynchronous datasets. On a synchronous dataset the model almost matches the results of the others.

Stec et al. [2018] expand the T-LSTM of Baytas et al. [2017] to multivariate, non-uniformly sampled time series, by introducing five different feature types. They introduce dense features for sequential features with high frequency, sparse

[3]An autoregressive model is a stochastic model where the output depends linearly on its own previous values as well as a stochastic error term.

feature for sequential features with low frequency, delta features related to the time between events, and two types of static features that apply to the sequences as a whole. The decay function is modified to handle an arbitrary number of delta features, not just the time elapsed since the last time step. The calculated decay factor is used to decay the short term component of the cell state as in Baytas et al. [2017]. In addition, the hidden state is split into distinct parts for long-term and short-term memory. Also each sparse feature has its own cell and hidden states that are kept unchanged between value updates. This is a result of the assumption that sparse features do not change between observations, so that only when a new observation arrives does the value change. This way features that are present frequently are handled differently from features that are present rarely when updating the cell states. Finally, the static features are applied to the last output of the LSTM cell as they apply to the sequence as a whole. They compare their model to the T-LSTM, and report getting better results on two data sets.

3.3 Summary

There seems to be a consensus in the reviewed works as to the main motivation behind using deep learning for anomaly detection. The deep learning approaches require little pre-processing. This means that less domain knowledge and less human labour is needed. Features that traditionally were handcrafted by experts can be substituted with features automatically extracted by the deep learning models.

To further summarize the reviewed literature, the literature review questions introduced on page 25 are revisited.

Q1 Has deep learning based anomaly detection been applied successfully to DTS systems?

A single paper was found that applies deep learning based anomaly detection to a DTS system monitoring pipelines, but few details of the constructed models were provided. It is thus hard to judge whether deep learning was successfully applied, but they do report getting better results than thresholding. The models are not described in a way to make them reproducible. They did report using the deep learning models to learn the normal behaviour of the time series, an approach that was used in many of the other reviewed works on anomaly detection, and is employed in this thesis as well.

Q2 What approaches exist to unsupervised anomaly detection in time series using deep learning?

Q3 What other deep learning approaches to anomaly detection in time series exist with other degrees of supervision and can they be transformed to be used in an unsupervised setting?

As little has been written directly on the subject of unsupervised anomaly detection, it seems natural to answer the second and third question together. Among the reviewed methods, the LSTM prediction model proposed by Malhotra et al. [2015] seems to be the most influential, with multiple applications in other reviewed works. The LSTM based autoencoder model has not seen the same wide application, which might be due to the fact that most researchers have been concerned with time series of a predictable nature.

Common to both methods proposed by Malhotra et al. is that they center around learning the normal behavior of the data, then use deviations from this to mark anomalous behaviour. The utility of this is motivated by normal behaviour data being readily available, while anomalous data often is scarce. In the export cables anomalous data is non-existent, but normal data is available. Thus methods based on modelling the normal behaviour are suitable.

While both methods proposed by Malhotra et al. are semi-supervised in their nature, they seem to be easily transferred to be unsupervised. Anomalous data is only used at the end to set appropriate threshold values, and this can be done with only normal data. An example of this was shown in Malhotra et al. [2016], where the threshold was set using only statistics of the normal data. This way the false positive rate can be controlled to a degree. The approach can also be applied to setting the threshold of the prediction based anomaly detection model.

Few fully supervised methods were proposed. This is probably a results of normal behaviour data being much more available than data of anomalous behavior in real-world scenarios.

Q4 How have asynchronous time series been handled in deep learning?

Traditionally, asynchronous time series in deep learning have been handled by having them resampled to the same frequency, as most models assume uniform sampling rates. Resampling has been argued to lead to loss of information from downsampling or high computational cost from upsampling. Most of the reviewed papers propose making changes to existing sequential deep learning models to accept asynchronous data, instead of resampling the inputs. The proposed models have mostly been alterations of the LSTM.

Some of the reviewed works address problems that are not relevant to the export cables. There are no categorical variables in the export cables as in Zhang et al. [2017], so there should be no need to symbolize the input. Baytas et al. [2017] and Stec et al. [2018] were concerned with taking the elapsed time into account.

The time between observations does not hold the same semantics in the export cables as in health care.

One of the simpler tricks reported in the reviewed works involve providing the time elapsed as a feature to a traditional deep learning model. This trick is attempted for the export cables. Also the representation of inputs preferred by Binkowski et al. [2018] is tested. The representation decomposes all the dimensions to a single dimension, with dimension markers and duration as additional features. Neil et al. [2016] were motivated by asynchronous time series produced by real world settings, and their proposed PLSTM will be tested for the asynchronous data from the export cable system.

3.4 Motivation

It has previously been explained that wind farm operators want to early detection of developing faults in the export cables. Currently, the only measure monitoring the studied export cables is crude thresholds. This measure is not able to provide early detection of developing faults. As has been the case in many of the reviewed works, plenty of data of the normal behaviour of the system is available. This motivates building upon the work of Malhotra et al. [2015] to model the normal behaviour of the export cable system. Deviations from the modelled normal behaviour can be used to detect anomalies.

Mathematically modelling the system of the DTS data and the environment surrounding the cable is not practical. This motivates using machine learning to learn a model of the normal behaviour. As in the reviewed works, there is a desire to avoid handcrafting features. This is believed to be difficult and time consuming, and will not yield results that can generalize to other domains or potentially even other cables. This motivates looking into deep learning methods to instead implicitly learn the complicated relationships directly from the raw data.

All the reviewed works on anomaly detection have looked at the temporal evolution of the variables of interest to find anomalous behaviour. The time dimension is believed to be of high importance also in the export cables. Further, the system is believed to be predictable, so a prediction based anomaly detection model might be appropriate. The electric current and sea temperature are believed to be big drivers of the evolution of the temperature in the cable, and these data sources are available to the analysis. This motivates first looking into how best to handle the asynchronicity in the time series that stem from combining the data sources related to the export cable. That a prediction based approach is believed to be suitable for the export cables, motivates focusing on asynchronous time series in deep learning prediction models.

The export cable data differs from the data addressed in the reviewed works

in that it also has a pronounced spatial dimension. The spatial sequences are highly correlated and might contain information that allows detecting anomalous behaviour. This motivates extending the general anomaly detection approach of modelling normal behaviour to be attempted in the spatial dimension as well. Finally it is believed that combining the temporal and spatial dimensions might enhance anomaly detection by looking at the temporal evolution of the spatial distribution of the DTS values.

Chapter 4

Experiments part 1: Asynchronous Time Series Data in Deep Learning Prediction Models

The next two chapters detail the two series of experiments conducted. The experiments of this chapter try to establish how best to handle the asynchronous time series data of the export cables in deep learning prediction models. This is done by exploring whether there are better ways to handle asynchronous time series data than resampling it to be of the same uniform sampling rate, which is the format assumed by most sequential deep learning models. The experiments are in direct connection to research question 1, which was defined on page 2. It is repeated here for convenience.

Research question 1 *How are the asynchronous time series of the export cable system best handled in sequential deep learning models?*

The asynchronous time series used in the experiments consists of the DTS value of a single segment, the electric current and the sea temperature. The electric current and sea temperature are uniformly sampled with sampling intervals of 10 minutes and one hour respectively. The sampling rate of the DTS system is uneven. The mean interval between the DTS samples is 1141 seconds, and the standard deviation is 176 seconds. Figure 4.1 shows a subset of the asynchronous time series with the different sampling rates of the component time series.

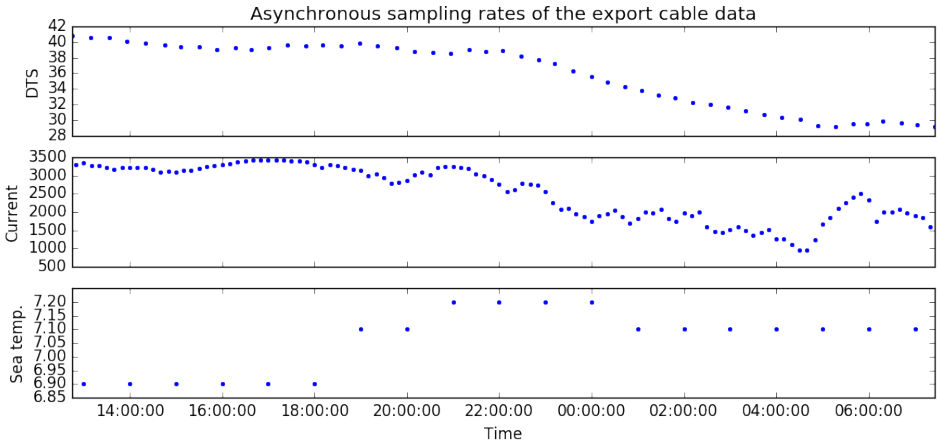


Figure 4.1: A subset of the asynchronous time series making up the export cable system. It consists of the DTS value of a segment of the export cable, the electric current through the cable and the sea temperature. Each blue dot represents an observation.

4.1 Experimental Plan

How best to handle asynchronous data is established for the export cables by comparing different techniques for handling asynchronous time series input when using RNNs to make predictions for the next temperature reading. Only a single segment of the cable is included in this series of experiments, together with the electric current and the sea temperature. Here, the temporal dimension is of interest. This leads to only regarding the DTS values of a single cable segment, as this allows studying the temporal dimension in isolation from the spatial dimension.

The particular problem of one step prediction is chosen for exploring how to handle asynchronous time series data because it is a simple and well-defined problem. No assumptions need to be made regarding the nature of anomalies or other aspects, as the time series data itself makes up both the inputs and the targets for training and evaluation. The problem gives results that make it easy to compare the tested models. Time series prediction is also a problem that intuitively should be affected by the assumptions made when handling the asynchronous data. If the model is to learn the physical relationships governing the temperature of a segment, the electric current and the surrounding environment, knowing the time passed between observations should make the relationships clearer when the time can vary.

A big advantage of reviewing time series prediction in this series of experiments, is that the implications of the results are directly applicable in the next series of experiments. There, prediction-based anomaly detection models are used for

anomaly detection in the export cables, and it is advantageous to have confidence in the prediction model being used in the anomaly detection models. Hopefully these first experiments will give general insight into how to treat asynchronous time series data in deep learning, but the end goal is finding insights that positively affect the downstream anomaly detection.

Section 3.2.3 shows that most of the contemporary approaches to deep learning with asynchronous data use LSTM-based models. For this reason, all approaches to handling asynchronous data are applied to LSTM-based models. Only regarding LSTM-based models has the benefit of keeping the models comparable, and reducing the scope of the experiments. The models are trained to predict the next temperature reading, given the last few temperature, electric current and sea temperature readings.

A few approaches are tested that resample the input data to make it synchronous. This resampling leads to loss of information. In the process of resampling, explicit assumptions need to be made about how the variables behave between observations. In the experiments the new values are interpolated linearly. Other approaches are tested that keep the input data at their sampled frequencies. For these, no assumptions need to be made about how variables act between observations. This should be an advantage, and thus it is expected that one of the latter approaches achieves the best results in the experiments.

4.2 Experimental Setup and Architectures

To ease and speed up training, the sliding window approach with stride 1 is used to construct samples, as described in section 2.7.1. Using a fixed size window to represent the state of the sequence is a simplification, but should hold in the export cables since the data is not expected to contain very long term dependencies. Physical properties govern the relationship between the cable temperature, the electric current though the cable and the sea temperature. This relationship may be of a lagged character, such that past values have an impact on future observations. These lagged relationships motivate using multiple past observations to make predictions. Multiple input steps might also expose information about the current physical properties of the surrounding environment of the cable, with regards to heat dissipation. This should improve predictions. The number of input steps to include in each sample is a hyperparameter that is decided by initial experiments.

Blocked cross validation with validation and test folds is used for training and evaluation, as described in section 2.7.2. At the beginning of each fold, each feature is z-normalized[1] based on the mean and standard deviation observed in

[1]That the data is z-normalized means that the population mean has been subtracted from

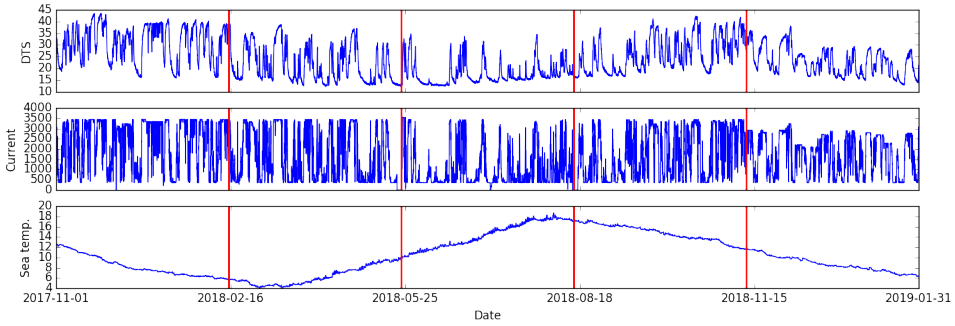


Figure 4.2: Data used in the first series of experiments. The red vertical lines show the division into 5 folds used for blocked cross validation.

the training data of the fold. This is done to put equal emphasis on each feature. Also the validation and test sets are z-normalized based on the mean and standard deviation of the training data.

When training, a maximum number of 1000 epochs is allowed with a patience of 50 epochs. This means that training terminates after 50 epochs with no improvement in validation accuracy or after a total of 1000 epochs, whichever comes first. The reported accuracy stems from the epoch with the best validation accuracy. All models are implemented in Tensorflow [Abadi et al., 2015]. Training is done in mini-batches of size 128.

This series of experiments was executed using 15 months of data from November 2017 out through January 2019. The segment 34327.4 meters from the shore was drawn randomly among all segments to be used in the experiments. Figure 4.2 shows the DTS values, the electric current and the sea temperature from this period. The red vertical lines show how the data was distributed into 5 sequential folds.

A number of different methods for handling asynchronous time series data were compared, and care was taken to make the comparison across the different methods as fair as possible. The same input samples were used by all models. Each input sample consisted of a fixed number of raw DTS values, with all the electric current and sea temperature observations in the same time range. As a result of this, the samples were of different duration. For the models requiring resampled inputs, each input sample was individually resampled. This way values outside the input sample could not affect the interpolated values. Most importantly, information about the target could not leak to the inputs values. All models were trained to make the same prediction, regardless of what resampling was applied to the inputs.

each sample, and the samples have been divided by the population standard deviation. This way the data has a mean of zero and a standard deviation of 1.

For all models, the target value to predict was the next raw measured DTS value. Figure 4.3 shows an input sample with corresponding target value.

The time passed from the last input to when the prediction was to be made for was also included as input to the models. This was to allow the models to handle the difference in time between the predictions. Samples that had prediction intervals larger than twice the median (2218s) or smaller than half the median (554.5s) were not included in training or evaluation.

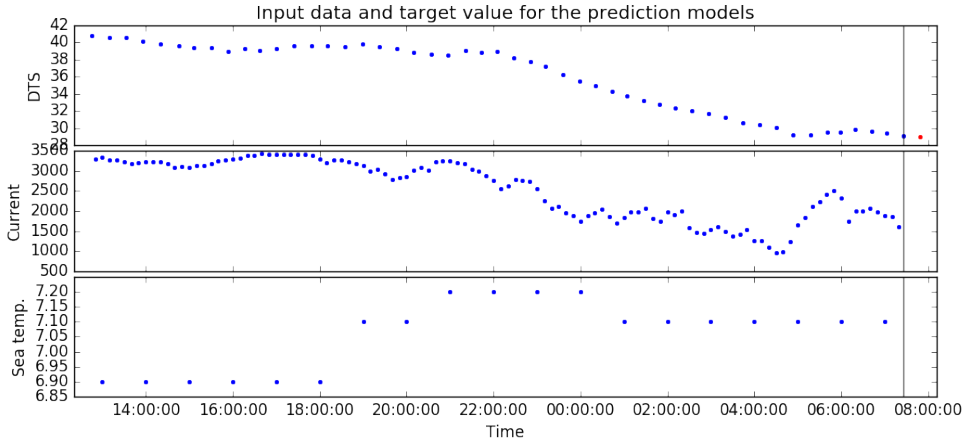


Figure 4.3: Input data and target value for time series prediction. The blue dots are the inputs at raw sampling rates. The one red dot to the right in the top plot is the DTS value to be predicted.

Before training the models of interest, some exploratory experiments were conducted to find reasonable hyperparameter settings. The specifics and full results of the exploratory experiments can be found in the appendix in section A.1.1. From this, the LSTM configuration was set to a single layer with 64 units. A learning rate of 0.01 was set to use with the Adam optimizer. Dropout was not used. The samples consisted of inputs with 50 input steps of DTS values with the sea temperature and electric current in the same time range.

The first models had the input data resampled to be synchronous. The data was then input directly to an LSTM cell. The final output of the LSTM was concatenated with the z-normalized time between the last input and the target. The result was forwarded to a fully connected layer with linear activation. The weights of the fully connected layer were initialized using a standard normal distribution, and the bias was initialized to 0. This architecture can be seen in figure 4.4. MSE was used to calculate the loss.

Table 4.1 summarizes the parameter settings common to all models. Now each model is presented with its specific input format and model architecture.

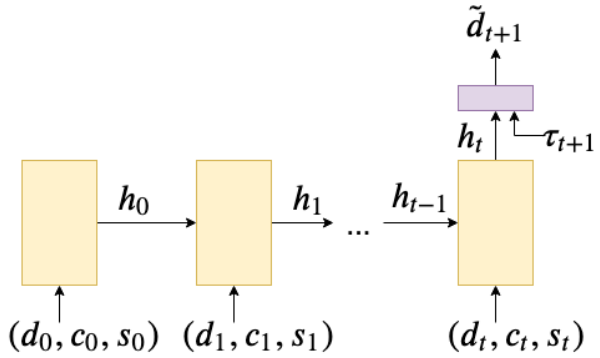


Figure 4.4: The base LSTM architecture used in the experiments. At each time step, the DTS value d_t, electric current c_t and sea temperature s_t is fed to the LSTM cell. The time until the prediction is concatenated to the final cell output, and the resulting vector is fed through a fully connected layer with linear activations to produce the prediction of the next DTS value $\tilde{d}_{t+1}$. Both the hidden and cell states are transferred between time steps, but only the hidden state is drawn.

Number of epochs	1000
Patience	50
Batch size	128
Learning rate	0.01
LSTM units	64
Optimizer	Adam

Table 4.1: Parameter settings for the first series of experiments

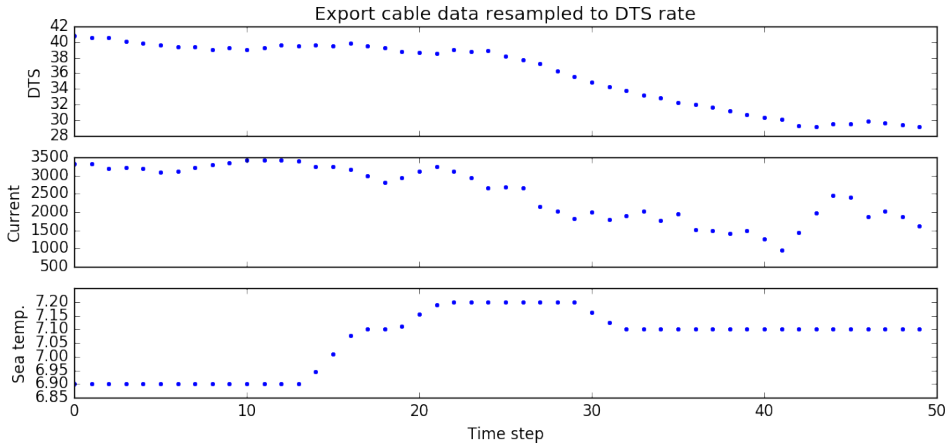


Figure 4.5: The input format used by model A. 50 time steps with the electric current and sea temperature resampled to fit the DTS sampling rate. Blue dots represent observations.

Model A: Values resampled to fit the rate of the DTS

The first model kept the DTS-value at its raw sampling rate. Both the electric current and sea temperature were resampled to fit, linearly interpolating values at the appropriate timesteps. As a consequence, the input to the model was synchronous but unevenly sampled, and there was no indication of the intervals between the observations. All the inputs to model A had 50 time steps of observations. Figure 4.5 shows an example of the resampled inputs. The basic LSTM architecture of figure 4.4 was used with the input.

Model B: Values resampled to fit the median DTS rate

For model B, the DTS value was resampled to make the observations uniformly sampled. The chosen sampling rate was 1109 seconds($\approx$ 18.5 minutes), which corresponds to the median sampling interval observed in the DTS data. Both the electric current and sea temperature were resampled to fit the resampled DTS value. Linear interpolation was used for the new values. Resampling to a uniform rate together with the uneven sampling rate of the DTS value led to inputs of different length, such that the longest had 69 time steps. The basic LSTM architecture of figure 4.4 was used.

Model C: Values resampled to fit the rate of the electric current

For model C, the electric current was kept at its raw 10-minute sampling rate. The DTS-value and sea temperature were resampled to fit. Again linear interpolation

was used to interpolate the new values. The resampling to a uniform rate together with the uneven sampling rate of the DTS value again lead to input samples of different lengths, where the longest had 126 input steps. The basic LSTM architecture of figure 4.4 was again used.

Model D: Raw temperature readings with interval as input

Model D had the same input as model A, but with the time passed between each DTS observation as an additional input feature. Thus the input consisted of the DTS value at its raw, uneven sampling rate, the electric current and sea temperature linearly interpolated to fit, and the time between each observation. As for model A, all samples had inputs with 50 input steps. The basic LSTM architecture of figure 4.4 was used, but with the time since the last observation τ_i as additional input to the LSTM cell at every time step.

Model E: All features collapsed into one dimension

Model E differs from the previously introduced models in that it allows all the measurements to be kept at their raw sampling rates. This is done by collapsing all the measured values into one dimension. Three other dimensions provide a one-hot encoding of what value is reported in each row. The last feature provides the time since the previous variable update. Figure 4.6 shows this format for a small input sample. Collapsing all features into one dimension made for longer samples, and they were again of different lengths. The longest sample had a length of 197 input steps. Normalization of each feature was done before combining the values. An architecture similar to that of figure 4.4 was used, but with the described input format.

Model F: Phased LSTMs

Also the phased LSTM allows the measurements to be kept at their raw sampling frequency. This was achieved by feeding each signal to its own PLSTM cell together with the timestamps of the measurements. These timestamps were constructed to be relative to the first temperature reading of each sample and were measured in seconds. The outputs from the PLSTMs were combined, and ultimately the z-normalized time between the last observation and the target was also input.

Three schemes for combining the three PLSTMs were applied to blocked cross validation with only validation sets. The most favourable of these was retrained using blocked cross validation with validation and test sets, in order to be compared to the other models.

The two first schemes accepted the raw inputs of each of the three signals. The input samples consisted of 50 input steps of DTS value and concurrent inputs

Time	DTS
00:35:13	37.147
00:55:05	37.041
01:17:26	37.421

Time	Sea T.
01:00:00	12.5

Time	Current
00:40:00	3374.0
00:50:00	3405.0
01:00:00	3406.0
01:10:00	3312.0

values	DTS	Sea T.	Current	interval
37.147	1	0	0	0
3374.000	0	0	1	287
3405.000	0	0	1	600
37.041	1	0	0	305
12.500	0	1	0	295
3406.000	0	0	1	0
3312.000	0	0	1	600
37.421	1	0	0	446

Figure 4.6: Input format used by model E. The values of the DTS, sea temperature and electric current are collapsed into one dimension. Three one-hot features mark what feature is being updated, and the last features counts seconds since the last value update.

steps of the sea temperature and electric current. The maximum number of input steps was 26 for the sea temperature and 126 for the electric current. Each signal was fed to its own PLSTM.

The first of the three schemes naively added the last outputs of the three PLSTMs, then concatenated the resulting vector with the time until predictions. The resulting vector was fed through a fully connected layer with a single output and linear activation.

The second scheme concatenated the three vectors containing the last outputs of each PLSTM to a long vector, and also included the time until prediction. The resulting vector was fed through a fully connected layer with sigmoid activation and 64 units. This result was again fed through another fully connected layer with only one output and linear activation.

The third scheme follows the approach described in Neil et al. [2016], and forwards all the PLSTM outputs to a final PLSTM. To allow this, the data for the third scheme is of a slightly different format than for the two other. To allow feeding the outputs to a final PLSTM, the input sequences need to be of the same length. The samples for all three signals were thus made to have the timestamps of the updates made to all of the signals. The last observed value was copied to fill all new timestamps for each signal. This was done in accordance with Neil's input

in an online forum[2] discussing how to train PLSTMs using popular framework implementations. The time gate should learn to ignore the copied values, and only regard the new updates.

With this input format, the three input signals were input into three different PLSTMs. The outputs at each time step were concatenated and input into a fourth PLSTM. The last output from this PLSTM was concatenated with the input time information and fed through a fully connected layer with a single output and linear activations to produce the prediction. A diagram of the architecture can be seen in figure 4.7.

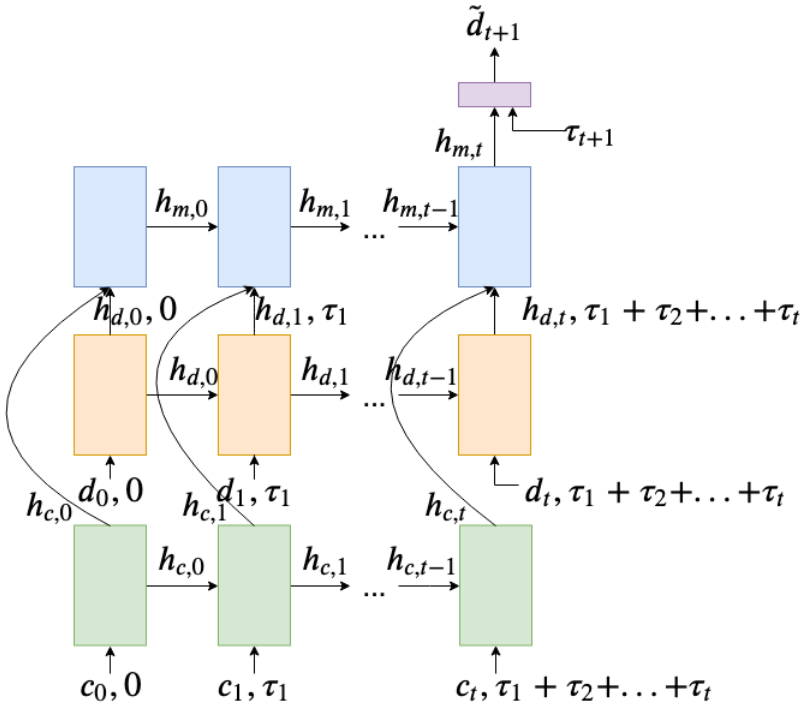


Figure 4.7: Phased LSTM architecture. The DTS value, electric current and sea temperature is fed to separate PLSTM cells at each time step. Only the cells for the DTS values and electric current are drawn. The same time input is presented to each cell. The top cell has as input the outputs from the bottom cells at each time step. Both the hidden state $\boldsymbol{h}$ and cell state $\boldsymbol{c}$ are transferred between timesteps, but only the hidden state is drawn.

[2] `https://www.reddit.com/r/MachineLearning/comments/5bmfw5/r_phased_lstm_accelerating_recurrent_network/`

The results of the initial PLSTM configurations using blocked cross validation can be seen in table A.5 in the appendix. Based on the results, the final approach with a fourth PLSTM merging the results was chosen for model F.

Further setup of the PLSTM is summarized in table 4.2 and was applied to all the tested units. r_{on} was initialized to 0.1, and the initial phase shift s was drawn uniformly between 0 and 0.1. The initial period τ was drawn from $exp(\mathcal{U}(1, 1000))$, where $\mathcal{U}$ denotes a uniform distribution. The three parameters controlling the time gate were learnable, and so they were changed from their initial values during the training process. The leak rate α was set to 0.001.

units	64
leak α	0.001
initial r_{on}	0.1
initial τ	$exp(\mathcal{U}(1, 1000))$
initial s	$\mathcal{U}(0, 0.1)$

Table 4.2: Parameters settings of the PLSTMs

Model G: Combined individual LSTMs

Model G also treated the raw observations of all three signals. The signals were each fed to their own 64 unit LSTM. The last output of each were concatenated, together with the time until prediction. The resulting vector was fed to a fully connected layer with sigmoid activation and 64 units. This result was again fed to another fully connected layer with only one output and linear activation. This is identical to one of the tested PLSTM approaches, but with the PLSTM cells substituted with regular LSTM cells. The architecture can be seen in figure 4.8.

Linear regression baseline

Finally, a linear regression baseline was implemented for comparison. The implementation `LinearRegression` from the module `linear_model` of sklearn [Pedregosa et al., 2011] was used. It needs fixed size inputs and was fed the same inputs as model D, namely the raw DTS values, the electric current and sea temperature resampled using linear interpolation to fit, and the time between observations. It was trained using the same cross validation scheme as the other models, thus using 3 folds for training and one for testing. The validation set does not have the same purpose for linear regression as for deep learning models. The model is fit analytically, so the validation data is not used for early stopping. Setting aside the validation set is however needed if the model is to be extended for anomaly detection.

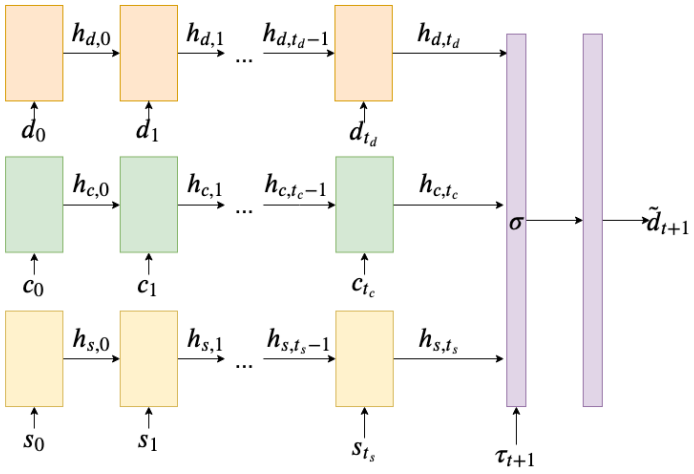


Figure 4.8: Model G architecture. Both the hidden state $\boldsymbol{h}$ and cell state $\boldsymbol{c}$ is transferred between timesteps, but only the hidden state is drawn.

4.3 Results and Discussion

This section presents the results of the experiments on asynchronous time series data in deep learning prediction, with a test of the statistical significance of the results. Following is a discussion of the results.

4.3.1 Experimental Results

Model	Train MSE	Val MSE	Test MSE
A: DTS rate	0.0451	0.0548	0.0673
B: DTS resampled	0.0520	0.0673	0.0787
C: El. current rate	0.0440	0.0574	0.0758
D: DTS rate + interval	0.0412	0.0575	0.0703
E: one dimension	0.1373	0.2230	0.3552
F: PLSTMs	**0.0407**	**0.0525**	**0.0611**
G: Combined LSTMs	0.0496	0.0563	0.0694
Linear regression	0.0550	0.0636	0.0636

Table 4.3: Averaged MSEs of the tested models for prediction with asynchronous time series data

Table 4.3 presents the results of the trained models. The reported MSEs are averaged over the in total 20 model versions trained for each test and validation fold combination.

The PLSTMs get the best averaged results for both the training, validation and test sets. On the training data, model D is very close behind. Also models C and A are close. Closest behind on the validation data is model A, while on the test set the linear regression model used as a baseline is closest. Model E, which collapses all value updates into one dimension, stands out as notably bad on all three data sets.

Quite a large gap can be observed in the training and testing MSEs of the neural networks. The linear regression model seems to have been better at generalizing, being the second worst model on the training data and second best on the test data. More discussion of this comes in the next section. First the statistical significance of the results is tested.

The Wilcoxon signed-rank test was applied to the MSEs of model F and each other models in turn, to see whether model F was the best model with statistical significance. The test was executed with a one-sided alternative hypothesis of model F having a smaller population mean. The paired samples used were the test MSEs of two models for the same combinations of test and validation folds. Thus 20 paired samples were input to the test. As four and four samples use the same test fold, the samples might invalidate the independence assumption of the test. No better suited test was however found, and it is believed to be the best way get some statistical insight.

The analysis found that model F was not statistically significantly better than model A nor the linear regression model at a significance level of 0.05. It was significantly better than the other tested models. The p-values of the statistical analysis are presented in table 4.4. These findings lead to a discussion of the initial results.

A	B	C	D	E	G	lin reg
0.07145	3.147e-05	0.004718	0.005344	9.537e-07	0.008591	**0.1305**

Table 4.4: P-values from the Wilcoxon signed-rank test with the alternative hypothesis that the MSEs of model F come from a distribution with a smaller mean than the MSEs of the other tested models.

4.3.2 Discussion

The statistical analysis shows that while model F was better than all other models on average, it was not statistically better than model A nor the linear regression model. This discussion will look at why that is the case. Table 4.5 shows the test results at finer granularity, where the test MSE is reported for each test fold averaged over the 4 corresponding validation folds. Tables A.6 and A.7 report the

validation and training MSEs respectively in the same way and can be found in the appendix.

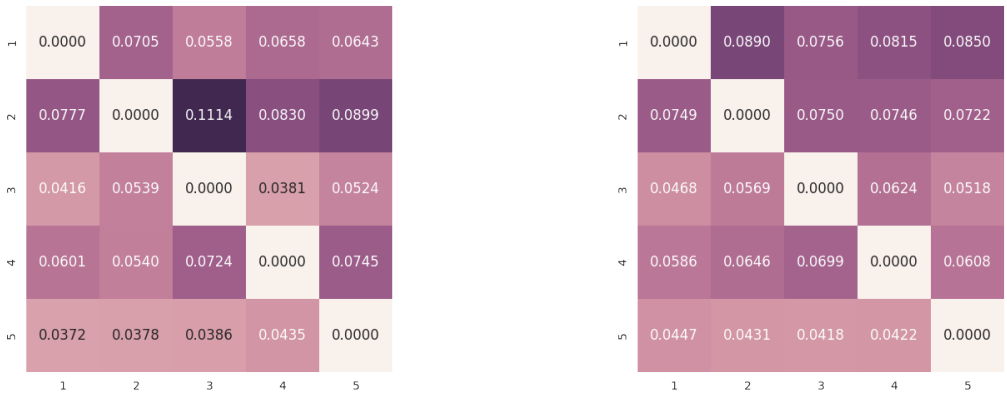
Model	test 1	test 2	test 3	test 4	test 5
A	0.0849	0.0966	0.0495	0.0651	0.0405
B	0.0874	0.1144	0.0576	0.0836	0.0507
C	0.0713	0.1087	0.0778	0.0787	0.0424
D	0.0657	0.1133	0.0515	0.0715	0.0496
E	1.0284	0.2091	0.2931	0.1539	0.0913
F	**0.0641**	0.0905	**0.0465**	0.0653	**0.0393**
G	0.0754	0.1060	0.0489	0.0687	0.0481
LR	0.0828	**0.0742**	0.0545	**0.0635**	0.0430

Table 4.5: Test MSE for different tested models for each test fold averaged over each validation fold.

From table 4.5 it can be observed that model F only performs best on the test set for 3 out of the 5 folds. On folds 2 and 4 the linear regression baseline beats all the deep learning models. This is also the case for the validation data. Model F beats linear regression on all folds for the training data. Many of the deep learning models show a better ability of fitting to the training data, but also a larger tendency to overfitting and not generalizing to the test data. It is of note that model D shows strong results on the training data, being ahead of model F on folds 2 and 5 and being almost equally good on fold 3.

The tables show that there are significant differences in the results given what fold is used for testing. This can be seen by inspecting the test results of model F. For fold 5 it has a test MSE of 0.0393, while for fold 2 it is 0.0905 which is more than the double.

Continuing this analysis, the tables in figure 4.9 show the test results for model F and the linear regression model at an even finer granularity, broken down to each test fold with each individual validation fold. Different test folds are on the rows, while the validation folds are in the columns. The cells report the MSE on the test set and are colored such that a darker color means a higher MSE. Linear regression seems more stable across the validation folds within each test fold (across the columns of each row), while model F and the other deep learning models are more unstable. Thus changing the validation fold for the same test fold has a bigger impact for the deep learning models. This might be a consequence of the deep learning model using the validation data during training for early stopping. Linear regression does not use the validation data at all for training, so the only effect is the change in the composition of the training data. This hints at potential overfitting to the validation data in the deep learning models. The



(a) PLSTM (b) Linear regression

Figure 4.9: Full test results for PLSTM and linear regression by test fold on the y-axis and validation fold on the x-axis. The numbers are the MSE on the test set. The squares are colored by magnitude of the MSE, so that low scores are bright and high scores are dark purple.

interested reader can find the full test results of models A, B, C, D, E and G in tables A.1, A.2, A.3, A.4, A.5 and A.6 respectively in the appendix.

There is some randomness present in the deep learning models due to random initialization of weights and random shuffling of samples during training. There is no randomness in linear regression. The randomness in deep learning could account for some of the variation across the folds, but the observed variation is believed to be too substantial and too similar across different models to be due to this randomness alone. The same combinations of test and validation folds are observed to stand out as bad for many of the models.

The boxplots of figure 4.10 show that the distributions of the electric current, the DTS value and the sea temperature vary substantially across the 5 folds. Further, the heat dissipation from the cable might have seasonal variances that are not captured by the input variables. This makes blocked cross validation problematic because the test and validation folds have data distributions that differ from each other and the training folds. This can lead to poor generalization.

The poor generalization can come from the models having to extrapolate when facing values in the test set that are more extreme than what have been observed during training. It might be that the linearity of the linear regression model handles the extrapolation better than the nonlinear deep learning models.

Another problem occurs when a validation set is used to guide early stopping, and this validation set has a distribution that is very different from the test set.

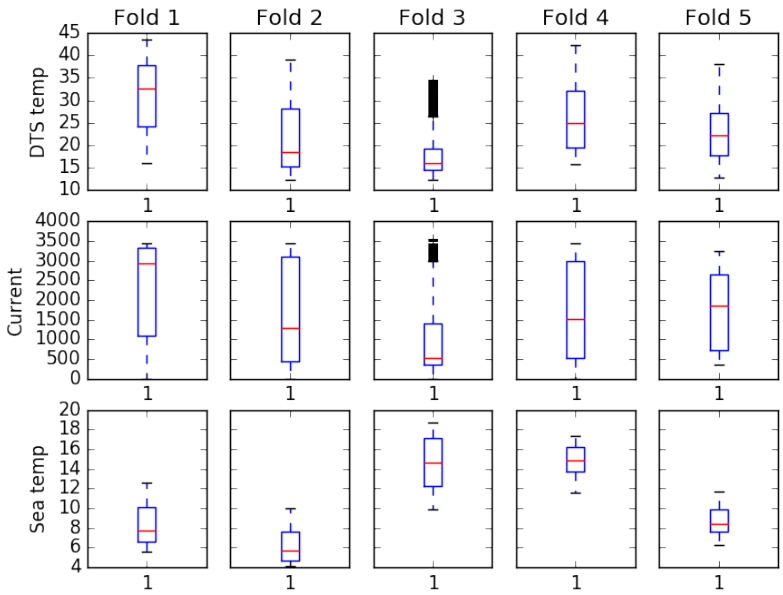


Figure 4.10: Boxplots of DTS value, electric current and sea temperature for each fold.

The different distributions might prefer different parameter settings, such that what best fits the validation set does not generalize to the test set. This way the deep learning models can overfit to the validation set.

4.4 Randomly Blocked Cross Validation

Based on the previous discussion, blocked cross validation seems unsuitable for seasonal data when few periods have been observed. This leads to the proposal of a training and evaluation scheme that tries to alleviate these issues. The proposed scheme involves making a much larger number n of sequential blocks, and constructing samples within each block. The blocks are then distributed randomly between the k folds, such that each of the k folds contains the samples of $\frac{n}{k}$ random blocks. An example of how this can look can be seen in figure 4.11 for $n = 60$ and $k = 5$, where each color represents a different fold. Only the DTS data is drawn to show the temporal division into the different folds.

The block size must be chosen so that the number of time steps in each block is at least the number of time steps needed in a training sample. Ideally the number of time steps will be significantly larger, as some samples are lost near the block borders when using a small stride for constructing samples. There is a trade-off

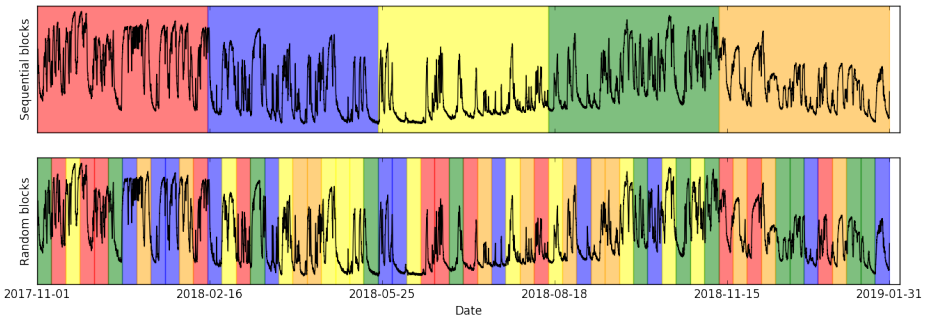


Figure 4.11: Fold division for sequentially blocked cross validation and for randomly blocked cross validation for the DTS data.

of having more samples or more blocks. With large amounts of data, finding a reasonable configuration should not be too challenging.

The discussion has made it clear than the training and evaluation scheme utilized might not have given a fair evaluation of the methods. It has also highlighted that linear regression seems more robust to extrapolation than the deep learning models. This series of experiments is wrapped up by a rerun of models A, F and linear regression with the proposed randomly blocked cross validation scheme. The hope is to now get statistically significant results, so that the model to extend to use for anomaly detection in the export cables can be established. Also model D is retrained with the new evaluation scheme, as model D showed strong results on the training set and might now generalize better.

The setup of the individual models was kept the same as in the first run, and is described in section 4.2. The only thing that has changed is the distribution of the data into the 5 folds used for cross validation with validation and test sets. Now the data is separated into 60 consecutive blocks, each with about 1 week of data. Samples were constructed from each block. The 60 blocks were randomly distributed among the 5 folds, such that each fold contained the samples of 12 random blocks of consecutive data. Then the same procedure for cross validation was followed, such that the data was normalized based on the data of the three training folds, the model was fit using the samples from the training folds, the samples of the validation fold were used for early stopping and the samples of the test fold were used only for evaluation in the very end. This was repeated for every combination of test and validation fold. The aim of this approach is to make the distribution within each fold more equal.

4.4.1 Experimental Results

The results of the rerun experiments averaged over test and validation folds can be seen in table 4.6. As before, model F has the best results for both the training, validation and test sets. The discrepancy between the training, validation and test sets is a lot smaller than it was using blocked cross validation with large sequential blocks.

Model	Train MSE	Val MSE	Test MSE
A: DTS rate	0.0352	0.0429	0.0461
D: DTS rate + interval	0.0341	0.0430	0.0450
F: PLSTM	**0.0336**	**0.0414**	**0.0448**
Linear regression	0.0562	0.0588	0.0588

Table 4.6: Averaged MSEs of the tested models for prediction with asynchronous time series data.

From statistical analysis of these results conducted in the same way as described in 4.3.2, model F is statistically better than model A and linear regression at a significance level of 0.05. It is however not statistically better than model D. The p-values of the analysis are presented in table 4.7.

A	D	Lin reg
0.03793	**0.2045**	2.861e-06

Table 4.7: P-values from the Wilcoxon signed-rank test with the alternative hypothesis that the MSEs of model F come from a distribution with a smaller mean than the MSEs of the other tested models.

4.4.2 Discussion

The boxplots in 4.12 show that the distributions of all three variables are pretty similar across the folds with the new fold division scheme. Making the blocks even smaller will make the distributions even more equal.

Figure 4.13 shows the full test results for models D and F respectively. In comparison to figure 4.9 from blocked cross validation with long sequential blocks, there is very little variability across the validation folds within the test folds. There is also a lot less variability across the test folds. Fold 3 stands out as getting markedly poor results for both deep learning models. This is especially true when used as the test set, but also when used for validation it systematically performs comparably bad. This is again probably due to differences in data distribution.

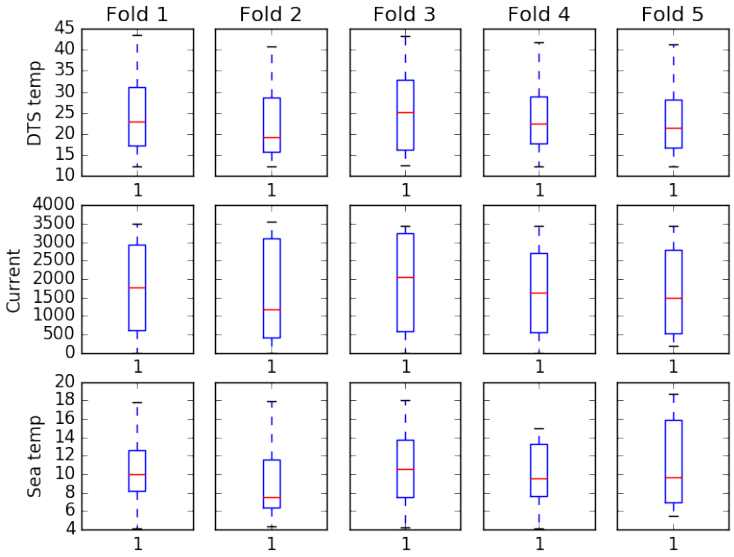


Figure 4.12: Boxplots for DTS value, electric current and sea temperature for each fold when using blocks containing 12 randomly distributed blocks.

From figure 4.11, fold 3 has three of its blocks next to each other, which might go towards explaining this. Using even smaller blocks would make this less likely to happen.

The full test results of the linear regression model can be seen in figure 4.14. It has even more even results across validation fold, and pretty even results across test fold as well. From the darker colors it can also be seen very clearly that it is outperformed by the deep learning models. For the interested reader, the full test results of model A can be found in figure A.7 in the appendix.

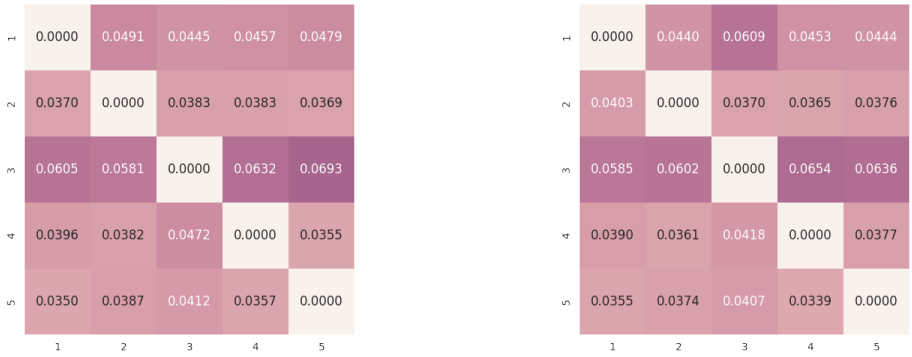
Thus it seems like the distributional problems of blocked cross validation with large sequential blocks for seasonal data have been alleviated by the proposed randomly blocked cross validation scheme.

4.5 Conclusion

The first series of experiment is wrapped up by revisiting the first research question.

Research question 1 *How are the asynchronous time series of the export cable system best handled in sequential deep learning models?*

Experiments have been conducted to explore how to treat the asynchronous time series of the export cable in time series prediction with deep learning mod-



Model D: DTS rate + interval Model F: PLSTM

Figure 4.13: MSEs for each test and validation fold combination for models D and F after retraining with randomly blocked cross validation. The test fold is on the y-axis and validation fold on the x-axis.

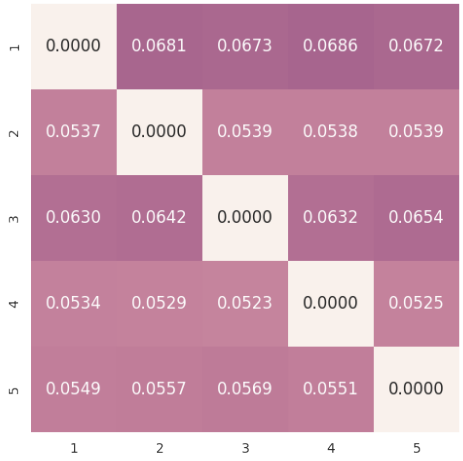


Figure 4.14: Full test results for linear regression after retraining with randomly blocked cross validation. Test fold is on the y-axis and validation fold on the x-axis.

els. Based on the results, there seems to be little to gain by using the much more involved phased LSTM, compared to simply resampling the data to be synchronous and using conventional LSTMs. There is no statistical difference in the performance of models F and D, and model D is chosen for the prediction based anomaly detection of next series of experiments. The ease of implementation of a conventional LSTM compared to the PLSTMs, and significantly higher speed of training contributes to making model D the preferred choice.

It is a surprising results that a model utilizing resampling is preferred. The resampling of the electric current and sea temperature was hypothesised to lead to loss of information that would hurt performance, and it is surprising that it did not significantly do so. The DTS value was however kept at its original sampling rate. It was the most important variable to learn, as it was the variable to be predicted. Still the electric current is hypothesised to be what drives the changes in the temperature, and thus having more information about the electric current was hypothesized to help predictions.

It seems like conventional LSTMs handle the uneven sampling rates of the DTS system well, at least when the time between samples is provided as an additional input feature. The sampling rate of the DTS system was however not terribly uneven. 50% of the intervals were between 16.5 and 21.4 minutes, and 95% were between 14.6 and 24.4 minutes. During the process it was wondered whether the samples were taken uniformly, but some post-processing lead to uneven timestamps. This was denied by the operator, so the uneven sampling rate is assumed to be real. This belief is strengthened by the fact that information about the sampling rate helps the deep learning model.

This wraps up the discussion of time series prediction with deep learning models with asynchronous input data. The focus now returns to anomaly detection, where the preferred model for time series prediction is used for anomaly detection and evaluated on its downstream anomaly detection results.

Chapter 5

Experiments Part 2: Anomaly Detection

In this chapter the focus returns to the main goal of this thesis, namely anomaly detection in the export cables. How best to exploit the spatial and temporal relationships in the DTS data to detect anomalies is explored. This is directly connected to the second research question, first introduced on page 2 and repeated here for convenience:

Research question 2 *Does exploiting relationships in both the temporal and spatial dimensions of the DTS-data give more robust anomaly detection models than only regarding one of the dimensions?*

Anomaly detection based on temporal and spatial relationships are first explored separately in sections 5.2 and 5.3. The lessens learned from these experiments guide how to construct models that exploit the relations in both dimensions. These are presented in section 5.4. The hypothesis is that utilizing both dimensions gives a more robust anomaly detector compared to looking at one of the dimensions in solitude. Exploring each dimension separately allows confirming or rejecting the hypothesis, and will lead to experience useful towards building the final combined models.

All models are trained only on normal data to learn the normal behaviour of the temperatures in the export cables, indirectly measured by the DTS system. This is done through prediction of future values or reconstruction of observed values from a smaller encoding. That the models have not been exposed to anomalous data during training is exploited for detecting anomalies. This is done by comparing model output to target values to get an error vector. The errors are expected to be larger for anomalous behaviour than for normal behavior, and this is used to detect anomalies. The approach is inspired by the results of Malhotra et al. [2015]

and Malhotra et al. [2016]. A set of synthetic faults are constructed and used to evaluate the models. First section 5.1 describes the experimental setup common to all the experiments of this chapter.

5.1 Experimental Setup

Data from 01.11.2017 to 03.04.2019 was used in the second series of experiments. The data was divided into 2-day blocks, and the blocks were randomly distributed into 5 folds as proposed in section 4.4. The four first folds contained 52 blocks and the last contained 50. The high number of blocks in the folds should lead to the data being evenly distributed.

As before, each experiment was repeated with different folds used for testing. Another fold was used for validation and the three remaining for training. Different from before was that only one designated validation fold was used with each test fold, instead of using each of the four other folds. This was the result of prohibitive training times. Training each model 20 times was not feasible. Each model was instead trained five times, once for each test fold. The subsequent fold was used for validation, such that test fold 1 used fold 2 for validation and test fold 5 used fold 1 for validation.

The data was z-normalized based on the statistics of the data from the training folds, and the models were trained using the normalized data from the training folds. The data from the validation fold was used for early stopping. Error vectors were obtained from applying the model to the validation data and comparing the outputs to the target values. These were used to fit the parameters of the anomaly score calculation, as described in section 5.1.2.

The data from the test fold was used for evaluation of the models. The blocks of the test fold were used to construct two types of synthetic faults. The blocks were also used as samples of normal behaviour. This allowed evaluating the amounts of both false negatives and false positives.

A fault sample thus consisted of 2 days of data. Inputs and targets were constructed from each fault using the sliding window approach with stride one and the sample format required by the respective models. The inputs were fed to the models. Each output obtained from the model was compared to its respective target value, and based on this an error vector was constructed. The error vector was used to calculate an anomaly score for the time step. The anomaly score for the whole fault sample was sat to the maximum anomaly score of the contained time steps. The same procedure was used to give anomaly scores to each sample of normal behaviour. The anomaly scores of the faults and normal samples were compared for each model to judge model skill. This was done by drawing ROC curves and calculating AUC scores.

5.1.1 Generation of Synthetic Faults

Synthetic faults in the export cable were constructed for evaluation of the anomaly detection models. They were generated in cooperation with a domain expert.

From the domain expert, two distinct situations describe the probable temporal development of future cable failures. The first situation is an instant fault. During an instant fault, the physical properties of the cable change rapidly. With the DTS system's sampling rate of about 18 minutes, this can lead to a visible temperature increase in the affected segments from one observation to the next. The other situation is a gradually developing fault that can be due to slow changes is the surroundings of a cable segment. The time frame of gradual faults will vary, and is difficult to predict. A gradual upwards drift of the temperature of the afflicted segments over a three month period is deemed prudent to detect by the domain expert.

When it comes to the spatial characteristics of future faults, they are believed to be of a local character. The implemented synthetic faults have a center segment affected with a temperature increase that dissipates to its four neighbouring segments on each side. This way the faults span about nine meters. Figure 5.1 shows how a 10% increase in the center segment dissipates to its neighbours. From the domain expert, the faults can have the temperature changes occurring for all values of the electric current.

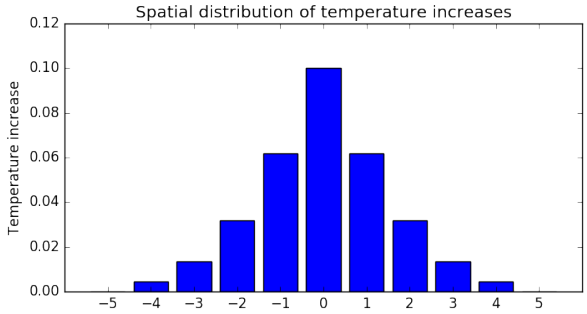


Figure 5.1: Spatial distribution of implemented faults. One segment is afflicted with a temperature increase that dissipates to four neighbouring segments on each side.

A memo of the characteristics of the failure scenarios written by the domain expert can be found on page 129 in the appendix. Some additional research scenarios are provided that are not implemented and evaluated in this thesis. They include shorter time periods for the gradual faults and different temperature changes. These scenarios are judged to be less difficult to detect than the implemented failure scenarios.

Instant Faults

The instant faults are implemented as a 10% increase in the temperature of the center afflicted segment between one time step and the next. The faults are constructed from blocks of two days of normal data. The blocks contain about 130 time steps of DTS values. 100 time steps of normal data are copied over to the fault. From time step 100 and out, the 10% increase is applied to the center segment, with smaller increases applied to its four neighbours on each side. An image of the temporal development of an instant fault can be seen in the bottom plot of figure 5.2. The plotted instant fault has been generated from the normal block in the plot above. The center afflicted segment and its five neighbours on each side are plotted. From time step 100 it can be seen that the temperatures of the affected segments vary more from their neighbours than before the fault occurred.

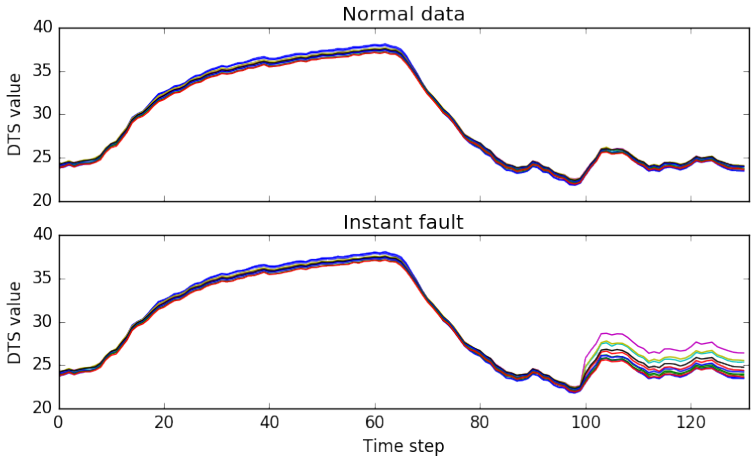


Figure 5.2: Temporal profile of a synthetic instant fault. 11 neighbouring segments are plotted, with an instant fault occurring at time step 100 for the bottom plot. The top plot shows the same segments in the normal data block used to generate the fault.

Quantiles of the normal distribution were used to get the shape of the fault requested by the domain expert. Increases of 6.17%, 3.171%, 1.34% and 0.46% were applied to the neighbours of the center afflicted segment in respective order of steps out from the center segment. The spatial temperature distribution after the inflicted fault can be seen in figure 5.3. The top image shows the temperature profile of a normal time step. The bottom plot shows the temperature profile for the same time step after an instant fault has been applied. Around segment 6000 the temperature has increased by 10% for the center afflicted segment. The change is drawn in red. Figure 5.4 shows the same time step, but has zoomed in on the afflicted segments.

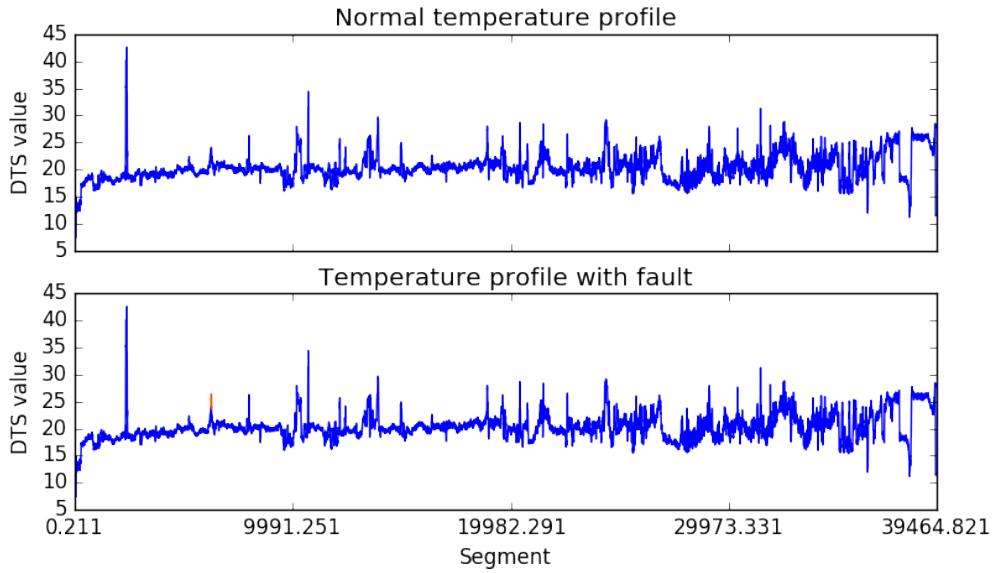


Figure 5.3: The spatial profile of the DTS values of the export cable. The top plot shows a normal profile for the export cable. The bottom plot shows the same profile with a synthetic fault drawn in red color around segment 6000.

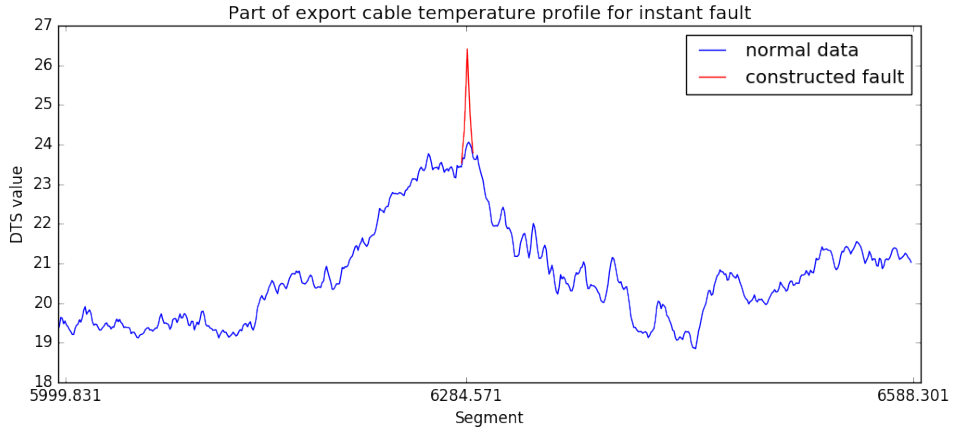


Figure 5.4: Closeup on the spatial DTS value profile of the export cable with a constructed instant fault. The red lines represent the change made to the normal frame to get the synthetic instant fault.

Gradual Faults

For the gradual faults, the temperature increase rises over 90 days from 0% to 10% for the center afflicted segment. This too is implemented with 2-day blocks as basis. The gradual drift of the temperature increase is assumed to be linear, and thus the temperature increase at the beginning and end of each block can be calculated. To simulate 90 days of increase, 45 2-day blocks are needed. These are randomly drawn and ordered from the available test blocks of each fold. The increases are applied to each block by calculating the appropriate temperature increase at every observed time step for the afflicted segments. The same quantiles from the normal distribution are used for the spatial distribution, such that the neighbours of the center segment have increases of 0.617, 0.317, 0.134 and 0.046 times that of the center segment.

Figure 5.5 shows three different stages of the gradual faults applied to the same normal block. Again 11 segments are plotted, with the middle one being the center afflicted segment. The top figure shows the two first days of the gradual increase. The temperature in the afflicted segment rises by 0.22%, which is not visible to the eye. The middle figure shows halfway through the increase. For this block, the temperature increase goes from 4.89% to 5.11%. The bottom plot shows the last two days of the gradual increase, where the temperature rises from 9.78% to 10% above the normal data.

Figure 5.6 shows a closeup of the spatial profile on the cable with the gradual increase at the same stages. At the end of the second day of the fault, the temperature increase is 0.22% which is hardly visible. After 46 days the increase is 5.11% and after 90 days the increase is 10%.

Construction

Faults were constructed for all the five test folds. Instant faults were constructed from every block of each test fold. Only 45 blocks were needed for the gradual faults. These were randomly drawn and ordered from the blocks of each test fold.

Faults were constructed for three segments. The rational for this will become clearer when the temporal models are described in section 5.2. The temporal models were only trained for a single segment, and were evaluated only on the faults constructed for that segment. The spatial and combined models were evaluated on the faults in all three segments. Reported experimental results are averaged over the three segments and 5 folds.

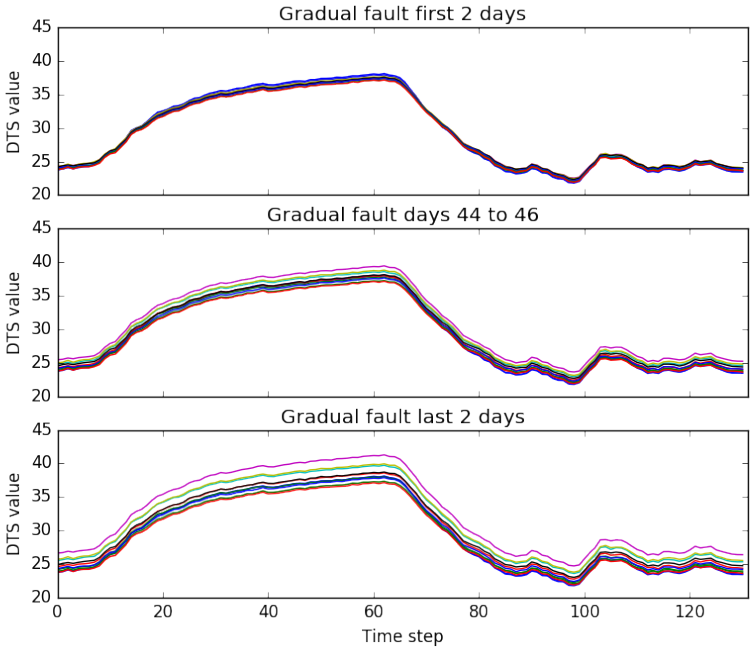


Figure 5.5: Temporal development at three stages of a synthetic gradual fault.

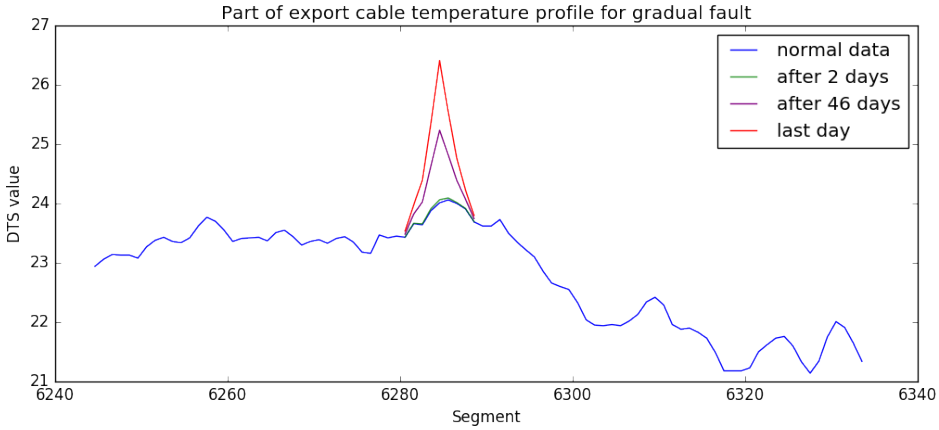


Figure 5.6: Closeup on the spatial DTS value profile of the export cable with different stages of gradual faults.

5.1.2 Anomaly Likelihood Score

The anomaly detection models produce error vectors. An observation is classified as anomalous or normal based on the characteristics of its error vector. The error vectors stemming from the normal data of the validation set are modelled to learn what error vectors are typical for normal behaviour, so that deviating error vectors can be used to detect anomalous behaviour.

This is achieved by using maximum likelihood estimation to fit a multivariate Gaussian distribution to the error vectors of the validation fold. Thus the mean vector of the distribution is set to the mean vector $\boldsymbol{\mu}$ of the validation errors, and the covariance matrix $\boldsymbol{\Sigma}$ of the distribution is set to the covariance matrix of the validation errors. This way both the typical magnitude of the errors and how they interact across terms for normal data is modelled.

$$f(\boldsymbol{x}) = \frac{e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}} \tag{5.1}$$

Equation 5.1 shows the density function of a multivariate Gaussian distribution. $\boldsymbol{x}$ and $\boldsymbol{\mu}$ are column vectors, where $\boldsymbol{\mu}$ is the mean vector of the distribution. The covariance matrix $\boldsymbol{\Sigma}$ must be positive definite, and $|\boldsymbol{\Sigma}|$ is its determinant. The only dependency on $\boldsymbol{x}$ is in the exponent. As there is a negative sign, the density decreases with an increase of the exponent. Thus a larger exponent will have smaller density than a smaller exponent. Based on these facts, an *anomaly likelihood score* is constructed as a proxy for the probability of an anomaly, as the determinant of the covariance matrix is unstable for large covariance matrices. Equation 5.2 shows the anomaly likelihood calculation.

$$a^{(i)} = (\boldsymbol{e}^{(i)} - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{e}^{(i)} - \boldsymbol{\mu}) \tag{5.2}$$

Calculating the expression is expensive and numerically unstable with large covariance matrices. Rewriting the expression as shown in equations 5.3, 5.4 and 5.5, allows using a differential equation solver instead of inverting the covariance matrix. This makes the calculation both more numerically stable and faster.

$$\boldsymbol{b}^{(i)} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{e}^{(i)} - \boldsymbol{\mu}) \tag{5.3}$$

$$\boldsymbol{\Sigma}\boldsymbol{b}^{(i)} = (\boldsymbol{e}^{(i)} - \boldsymbol{\mu}) \tag{5.4}$$

$$a^{(i)} = (\boldsymbol{e}^{(i)} - \boldsymbol{\mu})^T\boldsymbol{b}^{(i)} \tag{5.5}$$

The calculation can be sped up even further by calculating an LU factorization of the covariance matrix, and reusing the factorization to speed up solving the equation system of equation 5.4. This approach was utilized in this thesis, using

`lu_factor` and `lu_solve` from the `linalg` module of scikit-learn [Pedregosa et al., 2011].

5.2 Temporal Experiments

This section details the conducted experiments concerning the temporal dimension of the export cable data. The temporal dimension is explored by constructing models that only regard a single segment of the export cable, together with the electric current and the sea temperature. Multiple consecutive time steps are provided to the analysis to capture temporal trends. The models will have to learn how the temperature in a single segment normally develops, given the electric current through the cable and the surrounding sea temperature. Figure 5.7 shows the time series that are analysed by the temporal models.

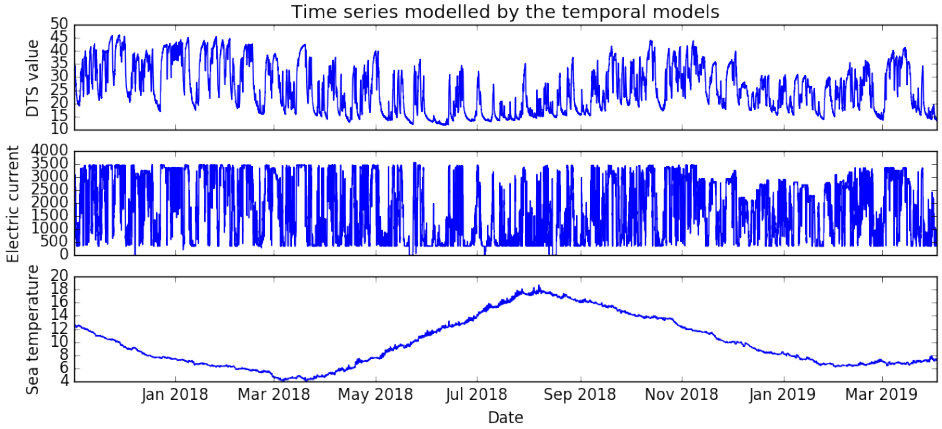


Figure 5.7: The DTS value for a single segment, electric current and sea temperature make up the time series that are modelled by the temporal models.

The goal of the experiments is to gain experience with temporal anomaly detection models. Having individual models for each of the 39516 segments does not scale well, and the hypothesis is that they are not powerful enough on their own. They should be able to pick up on the instant faults, as they will lead to abrupt changes from one time step to the next. The gradually developing faults might be more challenging, as the difference between each time step is minimal. The experiments result in experience towards deciding what model is best suited to expand with spatial information.

5.2.1 Experimental Plan

Methods based on predicting future values and reconstructing sequences are implemented. A prediction based model is hypothesized to perform best for the export cables, as discussed in section 3.3. LSTM based prediction models inspired by Malhotra et al. [2015] are implemented. Also LSTM autoencoder models inspired by Malhotra et al. [2016] are implemented, as well as a linear regression prediction model that will be used as a baseline.

LSTM Prediction

The prediction based LSTM models heavily lean on the results from the first series of experiments. Based on these, model D is used as a basis for anomaly detection. The model will make predictions for a number of time steps into the future. This way each time step will have a number of predictions made for it, and these predictions are compared to the observed value to produce an error vector. The error vector is used to calculate the anomaly score.

Models predicting both one and three steps ahead are implemented. It is hypothesised that one step is superior. The MSE will suffer from making predictions further into the future. It is guesswork to predict how the temperature behaves in an hour, when it is not known how the control variables behave. It is however not the MSE of the predictions that is the main concern, but rather the downstream anomaly detection results. The anomaly likelihood model should learn to accept larger errors in predictions further into the future.

Malhotra et al. [2015] predict both control variables and dependent variables when doing anomaly detection. The stated rationale is that it forces the network to learn normal usage patterns. If a control variable has changed between timesteps, this will likely lead to large prediction errors in the dependent variables, even when they behave normally given the change. When the control variable is also being predicted, the anomaly likelihood model will learn that large prediction errors in the dependent variables together with large errors in the control variable can be within normal behaviour.

In the export cables, the control variables are not of a binary on/off nature that results in abrupt changes, making this concern less relevant. There is also a concern of not wanting to find anomalies in the sea temperature or the electric current. Because of the unpredictable characteristics of the electric current, there is reason to believe that the network will not learn to successfully predict it. This fact should however be captured by the anomaly score calculation. It is still hypothesized that only predicting the temperature is more appropriate for the export cables. Models predicting only the DTS-reading as well as models predicting both the DTS-reading, the electric current and the sea temperature are implemented and

tested for the export cables.

LSTM Autoencoder

Also LSTM autoencoder models are implemented for anomaly detection. The models consist of two sub-models, first an LSTM encoder that takes a sequence as input and produces an encoding of it, then a decoder that takes the encoding as input and tries to reconstruct the input sequence. The models are trained jointly to reduce the mean squared reconstruction error obtained from comparing the reconstructed sequence to the input sequence. The reconstruction error vector is used to calculate the anomaly likelihood score.

Initial experiments examine what encoding should be produced by the encoder and passed to the decoder. Malhotra et al. [2016] report transferring the final state of the encoder to be the initial state of the decoder, but it is ambiguous what state is transferred. It can be only the cell state, only the hidden state or both. The papers Sutskever et al. [2014] and Srivastava et al. [2015] were also conferred. The first is also not clear on what state is transferred, while the latter report transferring both the hidden and cell states. All three variants are tested in an initial experiment to see what encoding is appropriate to use for the export cable dataset.

Models are implemented both for univariate and multivariate input. The univariate model only reconstructs the DTS readings, while the multivariate also reconstructs the electric current and sea temperature. In Malhotra et al. [2016] they only input univariate time series into their model. In the domains where they have multivariate inputs, they use the first principal component after applying principal component analysis. For a single segment of the export cable, the electric current and the sea temperature, only about 63% of the variance is explained by the first principal component. This is judged to be too little, and there are no technical reasons not to reconstruct all three signals. As for the prediction model, there is a concern that including the control variables will introduce anomalies that are due to the control variables not acting as expected, which is not what is wanted. There is however reason to believe the control variables to be instrumental in finding anomalies, as they decide what DTS temperature developments constitute as normal. Implementations are done both with and without the control variables to shed light on these concerns.

Linear Regression

The first series of experiments affirmed the importance of using a baseline to compare results to. The linear regression prediction model from the first series of experiments is also applied to anomaly detection, to be used as a baseline for

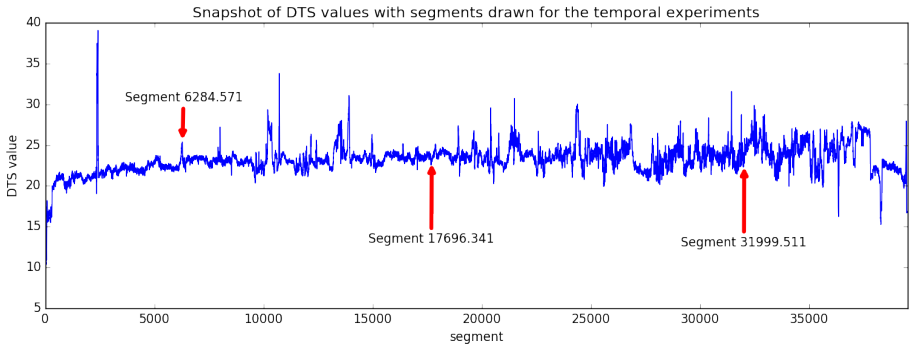


Figure 5.8: A snapshot of the DTS values in the cable at a random time step. The three segments drawn to use for the temporal experiments are marked by red arrows.

the temporal models. As for the LSTM prediction model, the predicted values are compared to observed values to produce an error vector that is used in the anomaly likelihood score calculation.

5.2.2 Experimental Setup and Architectures

The temporal models only cover a single segment of the cable, and it was not plausible to train models for all 39615 segments. Instead it was chosen to repeat the experiments for three random segments. This was thought to be more robust towards differences between segments than only using one. Three segments were randomly drawn, resulting in the segments located 6284.6, 17696.3 and 31999.5 meters from the shore. They are shown in figure 5.8.

Models were trained for each segment and faults of both types were constructed for evaluation. Training samples were constructed using the sliding window approach to make appropriate samples from each block of the training folds. More details follow on the implementations of the different temporal models.

LSTM Prediction

The LSTM prediction models are extensions of model D from the first series of experiments. The input thus consisted of the 50 last steps of DTS values for a segment, with electric current and sea temperature interpolated to fit as input. Also the time passed between samples was provided as input.

Four models were implemented that are differentiated by their target values:

LSTM p1 Predicts only the next DTS value

LSTM p3 Predicts the next 3 DTS values

LSTM p1 c Predicts the next observation of DTS value, electric current and sea temperature

LSTM p3 c Predicts the next 3 observations of DTS value, electric current and sea temperature

Figure 5.9 shows an example input sequence with the different target values.

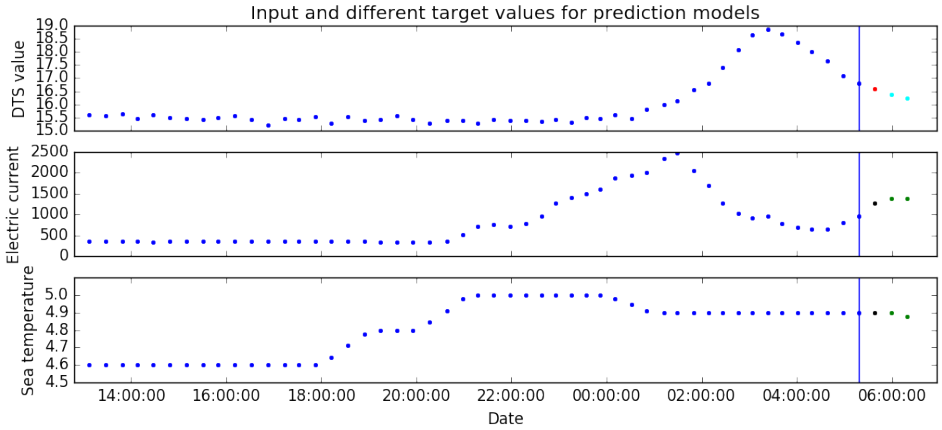


Figure 5.9: Inputs and targets for the prediction models. The dark blue dots represent the inputs to the prediction models. The blue vertical line shows the prediction limit. LSTM p1 predicts only the red dot. LSTM p3 predicts the red and turquoise dots. LSTM p1 c predicts the red and black dots, and LSTM p3 c predicts all the red, turquoise, black and green dots.

MSE was used as a loss function comparing the predicted values to the target values. The Adam optimizer was used to backpropagate the loss and update the weights with a learning rate of 0.01. The models only predicting the DTS temperature were 64 unit single layer LSTM. When also predicting the control variables, a 128 unit single layer LSTM was used. The setup further follows the setup of model D described in section 4.2.

LSTM Autoencoder

The LSTM autoencoders consisted of two single-layer LSTMs that were jointly optimized to reconstruct the input sequence. A conditioned LSTM autoencoder was used, so that the predicted values were fed back to the decoder cell during inference. The input sequence was reversed and fed to the encoder to build up the cell states. The final state of the encoder-LSTM was used as the initial state of the decoder-LSTM. The decoder was trained to reconstruct the sequence in the

reverse order of how it was fed to the encoder. All zero input was provided to the decoder LSTM in order to reconstruct the first value. For all the subsequent steps, the true target value of the previous step was provided as input during training. During inference, each predicted value was fed back to the model to make the next prediction. During both inference and training, the same weights and biases were applied to the LSTM output at each step to produce outputs of the right dimension. Figure 5.10 shows the inputs of the LSTM autoencoder at inference time.

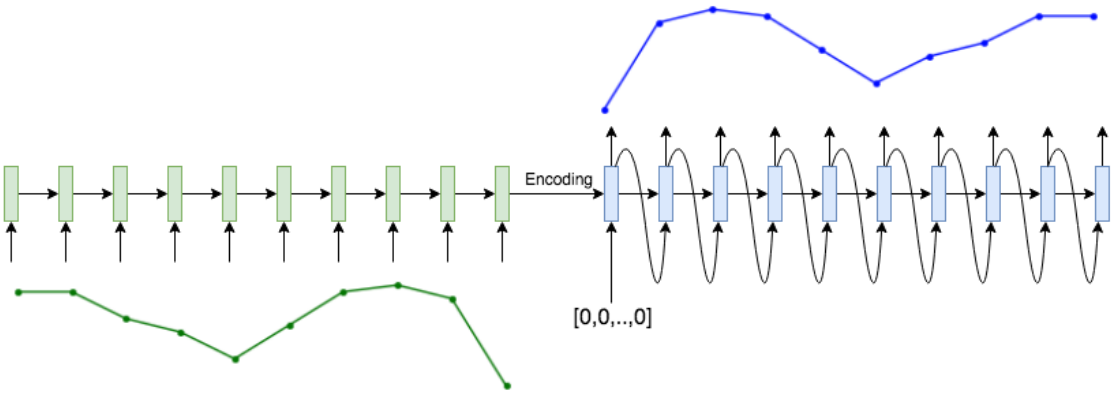


Figure 5.10: LSTM autoencoder at inference time. The green boxes represent the encoder LSTM cell, and the blue boxes the decoder LSTM cell. The sequence is reconstructed in the reverse order of the input.

The reconstructed sequence was compared to the input sequence, and the mean squared reconstruction error was used to calculate the loss. The Adam optimizer was used to backpropagate the loss and update the weights.

A window size of 50 time steps was used for the input sequences. The size is within a reasonable range when compared to the lengths of 30, 82, 208 and 500 used in the experiments reported in Malhotra et al. [2016]. There is no natural periodicity in the data to guide the window size, and setting the window size to 50 allow the LSTM-AE models same amount of context as the prediction models. Both the encoder and decoder LSTMs were single-layer LSTMs. When only reconstructing the DTS values, a 64 unit LSTM cell was used. When the electric current and sea temperature also were reconstructed, a 128 unit LSTM cell was used.

Initial experiments were conducted to decide what state to pass between encoder and decoder. Based on the results presented in table A.8 in the appendix, it was decided to only transfer the hidden state from the encoder to the decoder. The initial cell state of the decoder was set to all zeros. From the initial experiments the learning rate was set to 0.001. The reconstruction error obtained when com-

paring the reconstructed sequence to the input was used to calculate the anomaly likelihood score.

Linear Regression

The linear regression prediction model from the first series of experiments was also fit and applied to anomaly detection. The input was the same as for the LSTM prediction model, and thus consisted of 50 steps of DTS-temperature with the electric current and sea temperature interpolated to fit. Also the time between observations and the time until the prediction was provided as input. The target was the next DTS temperature. The prediction error from this one prediction was used for the anomaly likelihood calculation.

Table 5.1 summarizes the setup of all the temporal models.

Model	input dim	output dim	LSTM units	LR
LSTM p1	50×4	1×1	64	0.01
LSTM p3	50×4	3×1	64	0.01
LSTM p1 c	50×4	1×3	128	0.01
LSTM p3 c	50×4	3×3	128	0.01
LSTM AE	50×1	50×1	64	0.001
LSTM AE	50×3	50×3	128	0.001
Lin Reg	50×4	1×1	–	–

Table 5.1: Details of the setup of the temporal models. For the input and output dimensions, the first number describes the number of timesteps and the second number the number of variables at each each time step. 1 corresponds to DTS value only, 3 to DTS value, electric current and sea temperature and 4 to DTS value, electric current, sea temperature and the time since the last observation. LR is the learning rate used.

5.2.3 Experimental Results

The anomaly detection results of the temporal models are presented in table 5.2. The ROC curves for instant and gradual faults can be seen in figure 5.11. The results are presented as AUC scores averaged over the three segments and the five folds. To make the result tables easier to review, the best result in each column is bolded.

From table 5.2, regarding the temporal dimension gives close to perfect results for instant faults for several different models. For the gradual faults all models do poorly, hardly better than chance. This difference in performance of the temporal

Model	Instant faults	Gradual faults
LSTM p1	0.995	0.559
LSTM p3	0.981	0.553
LSTM p1 c	0.990	0.547
LSTM p3 c	0.902	0.532
LSTM AE1	0.842	**0.561**
LSTM AE3	0.715	0.529
Lin Reg	**0.998**	0.551

Table 5.2: Average AUC scores for the temporal models

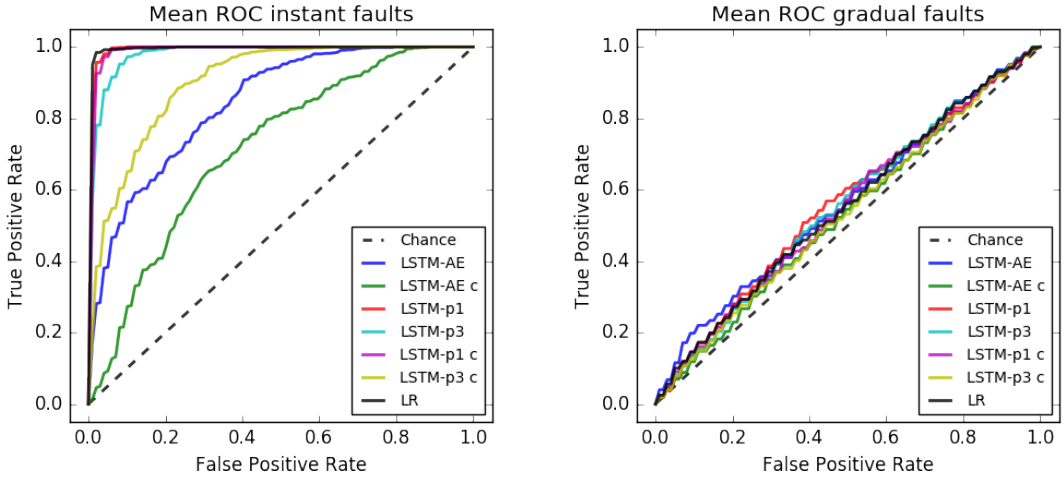


Figure 5.11: Mean ROC for the temporal models. Instant faults are on the left and gradual faults on the right.

models for the instant and gradual faults was partly expected. The gradual faults lead to very little relative difference from one time step to next, and thus at every time step the observed development might be well within normal behavior. The slight drift away from the neighbouring segments is of course not captured by the models only regarding the temporal dimension.

Again the Wilcoxon signed-rank test was used to test for statistical significance in the results. The paired samples in the anomaly detection experiments were two models trained for the same test fold and evaluated at the same segment. Thus 15 paired samples were input to the test when comparing two models. Again there is a concern of the independence of the samples, but the test is used as it is the best found alternative to getting statistical insight.

Linear regression surprisingly has the best results at detecting the instant faults. It is statistically better than all the other temporal models at a significance level of 0.05. The autoencoder only reconstructing the DTS values performs best at the gradual faults, but at a significance level of 0.05 it is not statistically better than LSTM-p1 or LSTM-p3. Tables A.9 and A.10 show the P-values of the statistical analysis, and can be found in the appendix.

For the instant faults, all prediction models do better than the autoencoder models. This corresponds to what was hypothesized due to the control variables being available. Among the autoencoders, the one only reconstructing the DTS values gets the best results. This too corresponds to the hypothesised result.

The best LSTM prediction model was the one only predicting the next DTS value, which also confirms the hypothesised result. Predicting multiple steps ahead, or also predicting the control variables, hurts anomaly detection performance. The additional variables should be more difficult to predict, but the resulting larger errors in the error vectors should be accommodated by the anomaly likelihood score calculation. However, the more difficult predictions get larger errors during training. The loss function used during training was MSE without any individual weighing of the different terms of the error vector. Larger errors in the less important additional variables steal focus away from making the best possible predictions for the next DTS value. This is believed to result in worse anomaly detection results. This might be improved by applying individual weights to the predicted variables in the loss calculation, to put more emphasis on optimizing the prediction of the next DTS value.

5.2.4 Discussion

This section investigates why the LSTM based prediction models were outperformed by linear regression in detecting anomalies. The implications of this result for constructing a model combining the temporal dimension with the spatial dimension are also discussed.

From the results of the first series of experiments, LSTM based prediction models should make more accurate predictions than the linear regression model. With the care taken when subdividing the data into folds, the models should not be extrapolating. The most successful LSTM prediction model is analysed to see where and why it fails under these conditions that should be pretty ideal. The analysis looks into the second fold trained and evaluated for segment 6285. Here the LSTM prediction model has an AUC score of 0.9889 for instant faults. The linear regression model is also analysed. It has a perfect AUC score of 1.0 for instant faults in the chosen fold and segment.

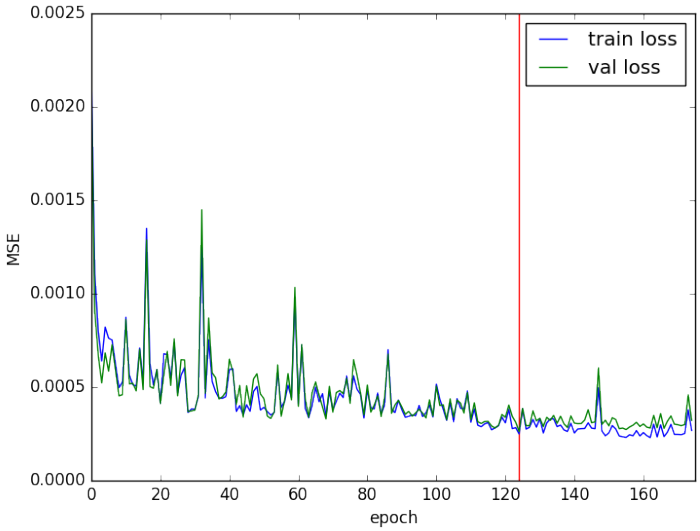


Figure 5.12: Training loss and validation loss plotted for the training procedure of the LSTM prediction model trained for fold 2, segment 6285. Red line marks minimum observed val loss.

The training procedure of the LSTM model is plotted in figure 5.12. While the training procedure might look unstable, it was found that the rather high learning rate of 0.01 hits lower validation errors than a smaller learning rate giving a more stable learning procedure. The lowest reached validation error is marked with a red line in the figure. It corresponds to validation MSE $2.62 \cdot 10^{-4}$, with training MSE $2.50 \cdot 10^{-4}$ and test MSE $3.10 \cdot 10^{-4}$. In comparison the linear regression model has training MSE $4.21 \cdot 10^{-4}$, and test MSE $4.37 \cdot 10^{-4}$. The LSTM thus makes significantly better predictions on average, confirming the results of first series of experiments. Still the linear regression model gets better anomaly detection results. It is not the average accuracy that is of importance to the anomaly detection, but rather if the errors being made for anomalous behaviour are sys-

tematically different from the errors being made for normal behaviour. Thus even if the LSTM on average makes better predictions, it might be less discriminatory towards the anomalies.

Figure 5.13 compares the anomaly scores of the linear regression and LSTM prediction models for the normal samples and corresponding constructed instant faults from fold 2, segent 6285. The blue dots represent the scores of normal samples, and the red dots at the same id represent the score of the fault constructed from the same block.

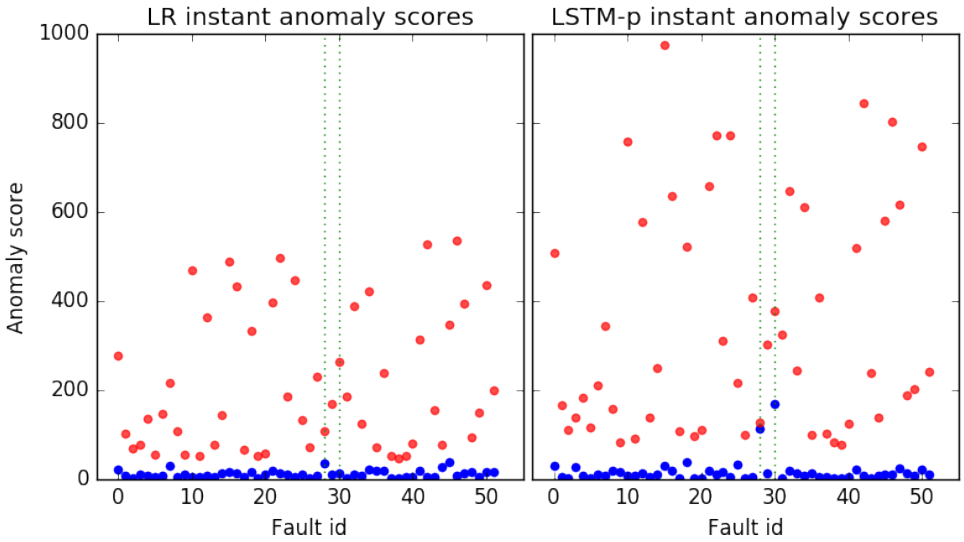


Figure 5.13: Linear regression prediction and LSTM prediction anomaly scores for the instant faults of segment 6285, fold 2. Red dots represent faults, blue dots the corresponding normal data the faults are constructed from. The green dotted lines at id 28 and 30 show the samples that the LSTM prediction model has problems with.

The LSTM notably has two outlying points among the normal points for blocks 28 and 30. Apart from these two points, the anomalous and normal points are clearly divided. The boundary between the classes seems wider for the LSTM than for linear regression, when the two outlying points are ignored. The difference in the mean scores of the normal and anomalous points is larger for the LSTM, with a difference of 332. For linear regression, the difference is 203. This suggests that the LSTM model would in fact have been better, where it not for these two samples of normal behaviour being classified as anomalous.

This raises the question of why the LSTM predicts these two points of normal behaviour so badly. The normal sequence of block 28 is plotted in figure 5.14 with the predictions made by the LSTM. The sea temperature is not plotted, as

it is not interesting to the discussion. Time step 60 has a bad prediction and is marked by an orange dotted line. Here the LSTM model predicts a substantial increase in the temperature that does not occur. This gives one very high anomaly score, which gives a high anomaly score to the whole sample. While it is difficult to interpret the output of deep learning models, it looks as if the bad prediction happens at the first time step where the current decreases after a quick increase. One can imagine that the model expected the current to continue its increase, and thus expected the temperature to rise sharply. Also the outlying normal sample of block 30 has a large error following a quick change in the electric current, as can be seen in figure in 5.15.

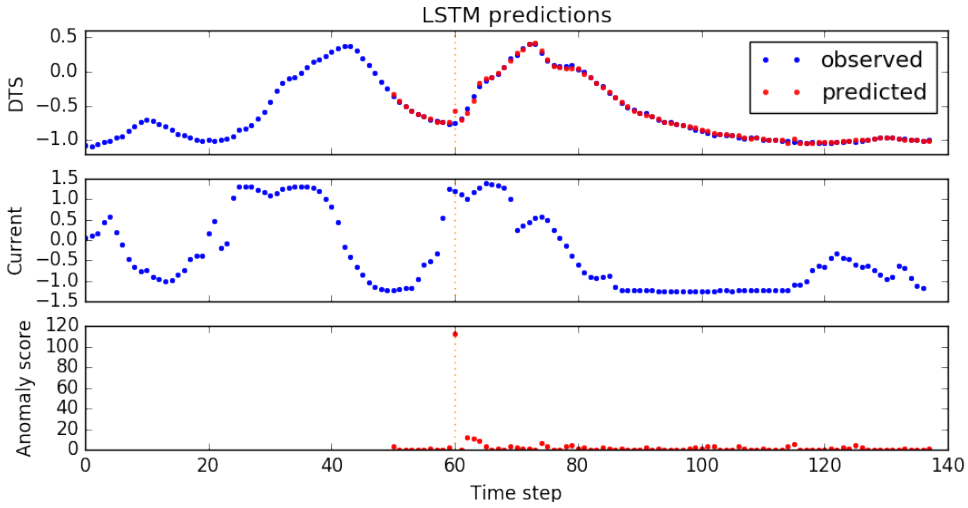


Figure 5.14: LSTM prediction for the normal sample of block 28. The orange dotted line shows time step with largest error. The data is normalized.

Figure 5.16 plots predictions for block 28 for the linear regression model. It is not thrown by the sudden change in the direction of the current. It does also give a large scores when the current shifts around time step 75, but this score is still below all the anomalous scores. Figure 5.17 shows the linear regression predictions for the normal data of block 30. Here all the scores are low.

It can be concluded that while the LSTM makes good predictions on average, it is not reliable enough to beat linear regression at anomaly detection. The wide border between the anomalous and normal points, apart from the outliers, motivates looking into making its predictions more stable.

From the plots of the predictions, the observations of the electric current seem a lot less stable than the observations of the DTS temperature. A lot can happen to the electric current in the $\approx$ 18 minutes between each sample. This uncertainty should have been captured by the models, but the LSTM prediction model has not

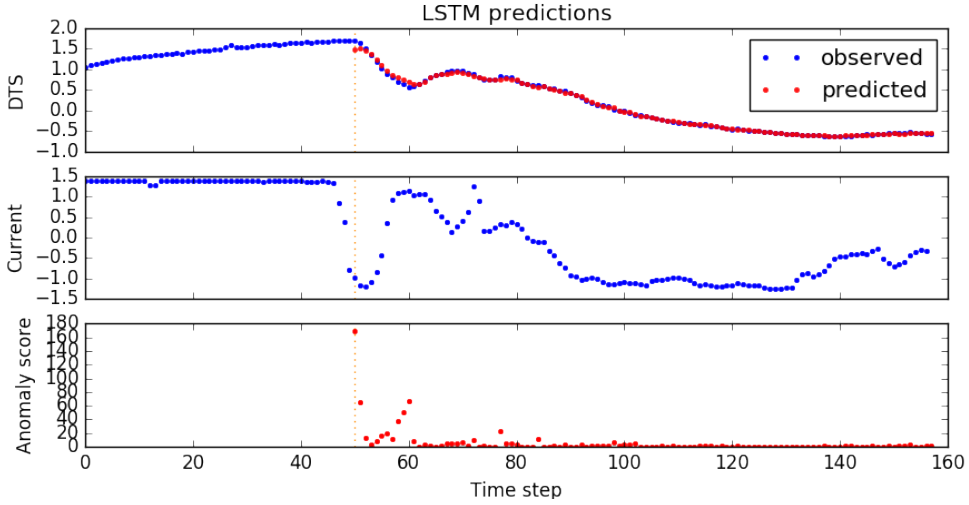


Figure 5.15: LSTM predictions for the normal samples of block 30. The orange dotted line shows time step with largest error. The data is normalized.

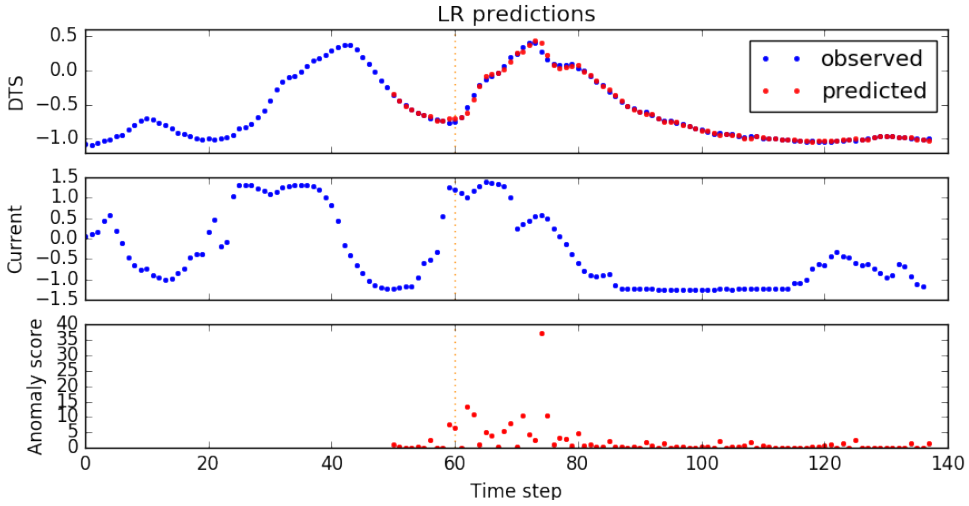


Figure 5.16: Linear regression predictions for the normal samples of block 28. The orange dotted line shows where it goes wrong for the LSTM. The data is normalized.

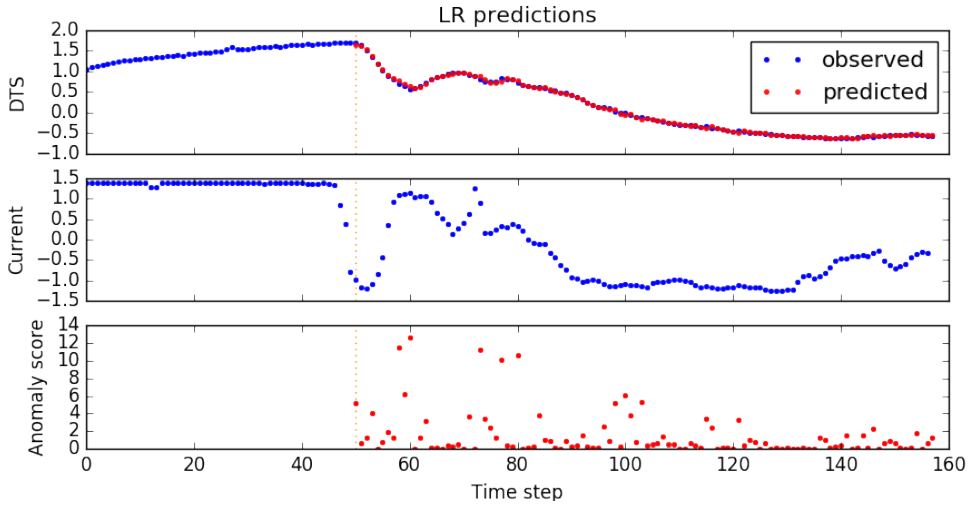


Figure 5.17: Linear regression prediction for the normal data of block 30. The orange dotted line shows where it goes wrong for the LSTM. The data is normalized.

been very successful. Predicting the control variables in addition could catch the prediction errors due to the electric current not behaving as expected. This was however attempted, but resulted in worse anomaly detection results. In figure 5.18, the anomaly scores of the LSTM also predicting the control variables for the next time step are compared to scores of the LSTM only predicting the DTS value. The outliers are not a problem in the right plot, but also many of the anomalous points have a lower score. Thus predicting the control variables is not the solution to making the LSTM prediction model better at distinguishing normal and anomalous behaviour.

The first series of experiments found that resampling the electric current to fit the temperature did not hurt predictions much on average. Including more information about the electric current might still lead to more stable predictions. This can be done in two ways, that can also be combined. The first way is to change the input format to accept the raw sampling rate of the electric current. This can be achieved by retrying the resampling scheme of fitting the temperatures to the electric current observations, though might not be preferred as information is lost about the DTS value. Instead, the phased LSTM can be applied to include all the current observations. Another way to allow the model more information about the electric current is to include all the electric current observations up to and including the time of the observation of temperature that is to be predicted. Figure 5.19 shows the proposed input. The two electric current observations between the blue and red lines were not included in the input to the trained models, but might improve predictions. This can be implemented without any major changes to the

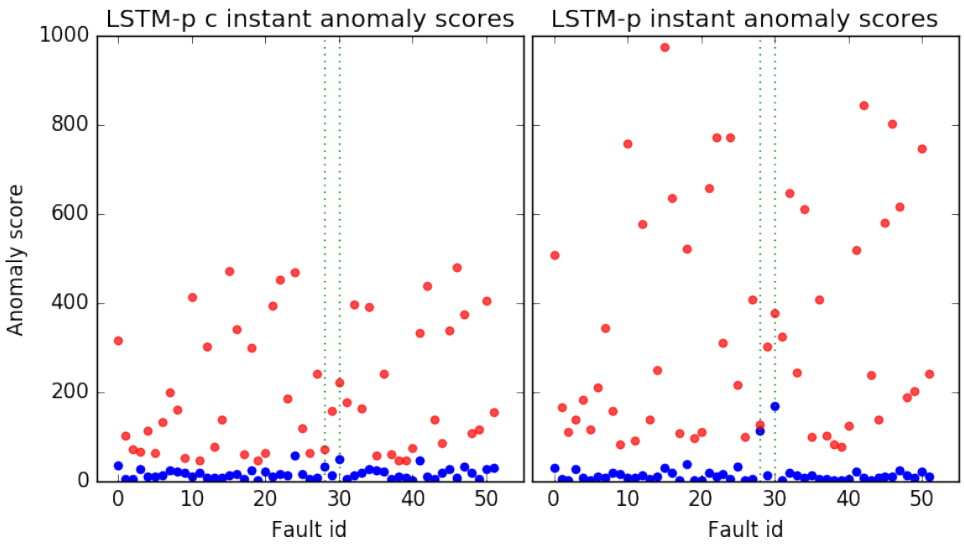


Figure 5.18: LSTM prediction with and without predicting the control variables. Anomaly scores for instant faults of segment 6285, test fold 2. Red dots represent faults, blue dots the corresponding normal data the faults are constructed from. Index 28 and 30 are marked with dotted green line to show outliers in plot on the right.

architecture. The electric current could be concatenated to the last LSTM output in the same manner as the time until the next prediction.

The sampling rates might be too low for the LSTM to make reliable predictions, even if more information about the electric current is included. The linear regression model did however get pretty good results for the instant faults, but the margin dividing the normal and anomalous points was not very wide. The next section will show that spatial models get even better results for the instant faults. It thus seems to be the case that the temporal dimension is not critical for detecting anomalies, contrary to what was hypothesised.

The anomaly scores for gradual faults are also plotted for the linear regression model and the LSTM prediction model. They can be seen in figure 5.20. The temperature increase is 0 to the very left and rises gradually to a 10% increase to the right. As has already been established by the average AUC score, none of the models are very skillful. Any differences between the scores of the normal samples and the faults are dwarfed by the differences in scores within the normal samples.

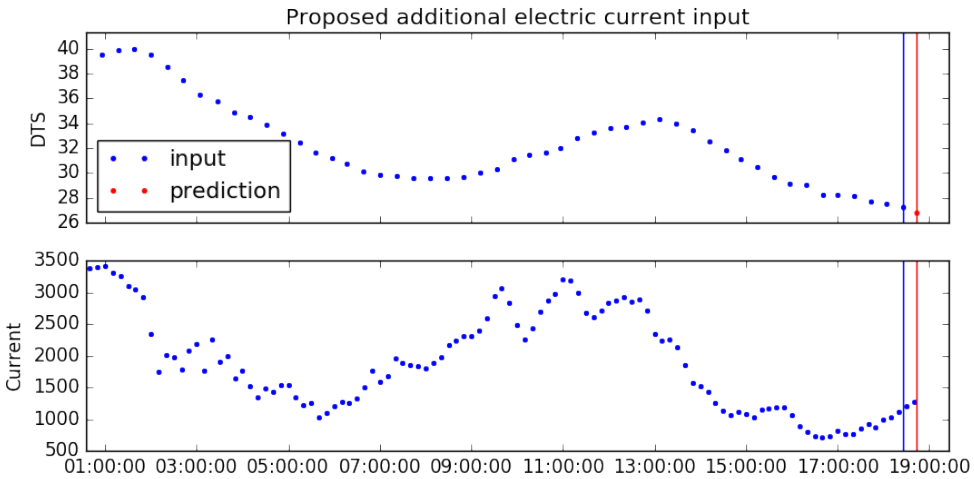


Figure 5.19: Proposed model input with more information about the electric current

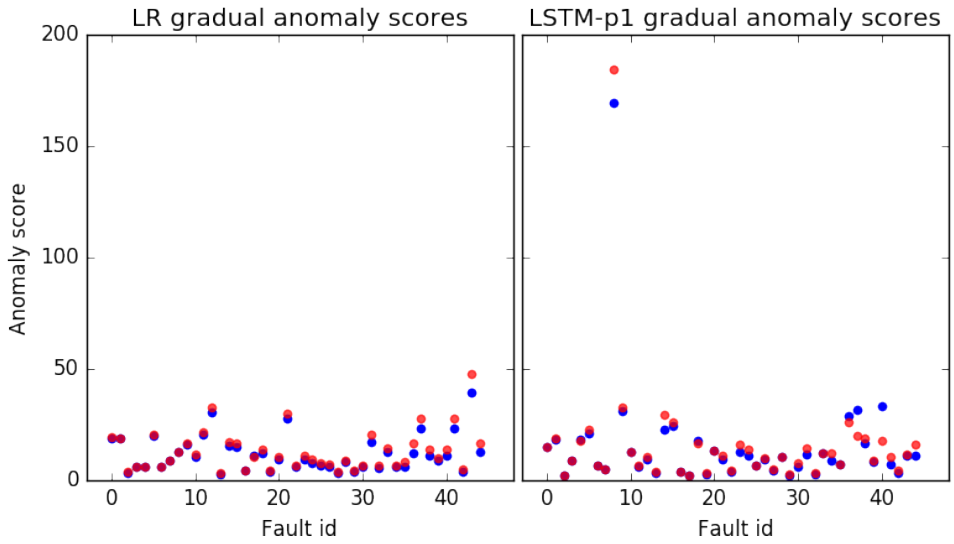


Figure 5.20: Linear regression and LSTM anomaly scores for the gradual faults of segment 6285, fold 2.

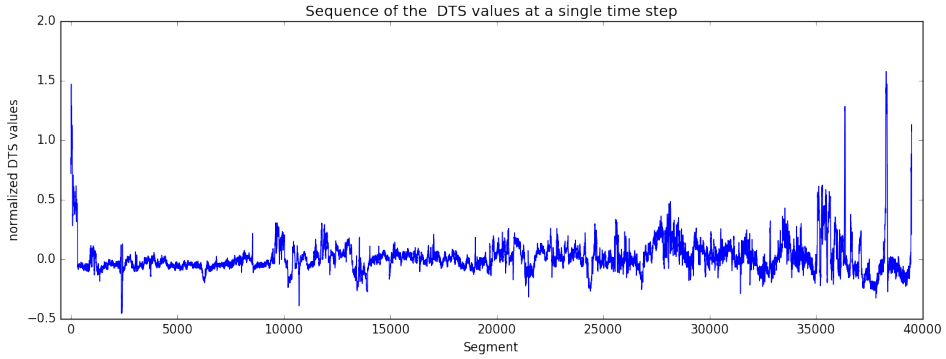


Figure 5.21: A single timestep of normalized DTS values is the input to the spatial models.

5.3 Spatial Experiments

This section presents the experiments executed on the spatial dimension of the export cable data. The models of this section have a single time step of all the DTS data as input, and might have more success at detecting some segments drifting from the rest. While the temporal models were said not to be scalable, the spatial models do not need to be scaled up to be deployed and used, as they already handle the whole cable. Thus the models of the spatial experiments can be used for anomaly detection in the export cables on their own should they be successful. Figure 5.21 shows an example sequence of DTS data that constitutes the input to the spatial models.

5.3.1 Experimental Plan

The general approach to anomaly detection in the spatial dimension is to construct a more compact encoding of the sequence of DTS values at a single time step, then reconstruct the original sequence from this encoding. The error in the reconstruction is used to calculate the anomaly score.

The spatial experiments have two main objectives. The first is find a good model for anomaly detection in the export cables. A lot of data is available in the spatial dimension, which might be sufficient to find both instant and gradually developing faults. A sub-goal of the experiments is to find a useful, smaller representation of the data that can be fed to a temporal model for enhanced anomaly detection.

The spatial models are evaluated on the faults constructed for the three segments of the temporal experiments. This allows comparing the spatial and temporal models across the experiments.

Autoencoders

A number of neural network autoencoder models are implemented. They are evaluated both on their ability to find anomalies, but also on their intermediate ability to reconstruct normal sequences. First some fully connected architectures are attempted. Also some more complex architectures based on convolutional layers are implemented.

Principal Components Analysis (PCA)

PCA is implemented as a baseline for the spatial models. PCA is fit to the data, and different numbers of components are kept and used to encode and reconstruct the sequence. Again both anomaly detection based on reconstruction error, and how good the models reconstruct normal input is evaluated.

5.3.2 Experimental Setup and Architectures

This section presents more detailed setup of the spatial models. The analysis of the spatial dimension led to a big increase in the amount of data to handle, as the model input now consisted of 39516 values. This led to some computational challenges that had to be overcome.

Autoencoders

Initially a fully connected, undercomplete autoencoder was planned as a baseline spatial deep learning model. This architecture enforced an enormous amount of weights and gradients to be computed, and turned out not to be feasible to train. As a result of this, fully connected architectures were skipped from further consideration.

In hindsight, a fully connected architecture seems excessive. It should not be necessary for every segment in the cable to be connected to every other segment in the whole of the cable. The spatial locality in the cable should be captured by a convolutional autoencoder (CAE) [Masci et al., 2011]. Local connections and shared weights reduced the number of parameters to fit, making training plausible.

The encoder consisted of four convolutional layers, each followed by a max pooling layer reducing the encoding size. The decoder consisted of four transposed convolutional layers each followed by an upsamling layer. It increased the size of the encoding at the same rate as it had been decreased by the encoder.

In the convolutional layers 1D convolutions were computed. The stride was set to one, and padding was used to maintain the encoding size. Bias was added to the computed values with one bias per output channel, and the result was passed

through a sigmoid activation function. Then max pooling was applied to reduce the dimension of the representation.

In the transposed convolutional layers of the decoder, transposed 1D convolutions were computed. Bias was added with one bias per output channel, then sigmoid activation was applied. The upsampling layers upsampled to the desired length by linearly interpolating values. The channels of the final layer were combined by multiplication with weights and addition of a bias. The encoder and decoder did not share any weights between them. Figure 5.22 shows a CAE architecture with 2 convolutional and transposed convolutional layers.

Multiple channel, width and stride configurations were planned to train, and the configuration with the largest encoding size was trained first. It used a width of 5 for the convolutions in all four layers. All max pooling layers used a width and stride of 4. The first convolutional layer took the DTS values as input, and thus had a single input channel. All layers had 30 output channels. The details of the encoder are presented in table 5.3. This configuration led to an encoding of size 4650 at the narrowest point, which is about 12% of the original length of the cable. The decoder increased the encoding size at the rate it was decreased by the encoder. All of the layers had 30 output channels, and the 30 channels of the final layer were combined by multiplication with weights and adding a bias.

conv	max p.	out channels	channel width	encoding size
		1	39 516	39 516
5,1	4,4	30	9879	296 370
5,1	4,4	30	2470	74 000
5,1	4,4	30	618	18 540
5,1	4,4	30	155	4650

Table 5.3: CAE encoder setup. The first row shows the input size. The first column presents the width and stride used by convolutional layers, while the 2nd column presents the width and stride used by the succeeding max pooling layers. The encoding size is given by channel width × number of out channels.

Another configuration was attempted that produced a smaller encoding. Again convolutions with a width of 5 were used, but the max pooling layer used a width and stride of 5 for the last layer and 6 for the rest. The final layer of the encoder had a reduced number of 4 output channels. This lead to an encoding size of 148. The configuration is summarized in table 5.4. The first layer of the decoder had 4 input channels. All decoder layers produced 30 output channels, and these were again combined for the last layer by multiplication with weights and addition of bias. After training for the first fold, the evaluation came out terrible, and further training was put on hold. Due to the impressing results of the much simpler

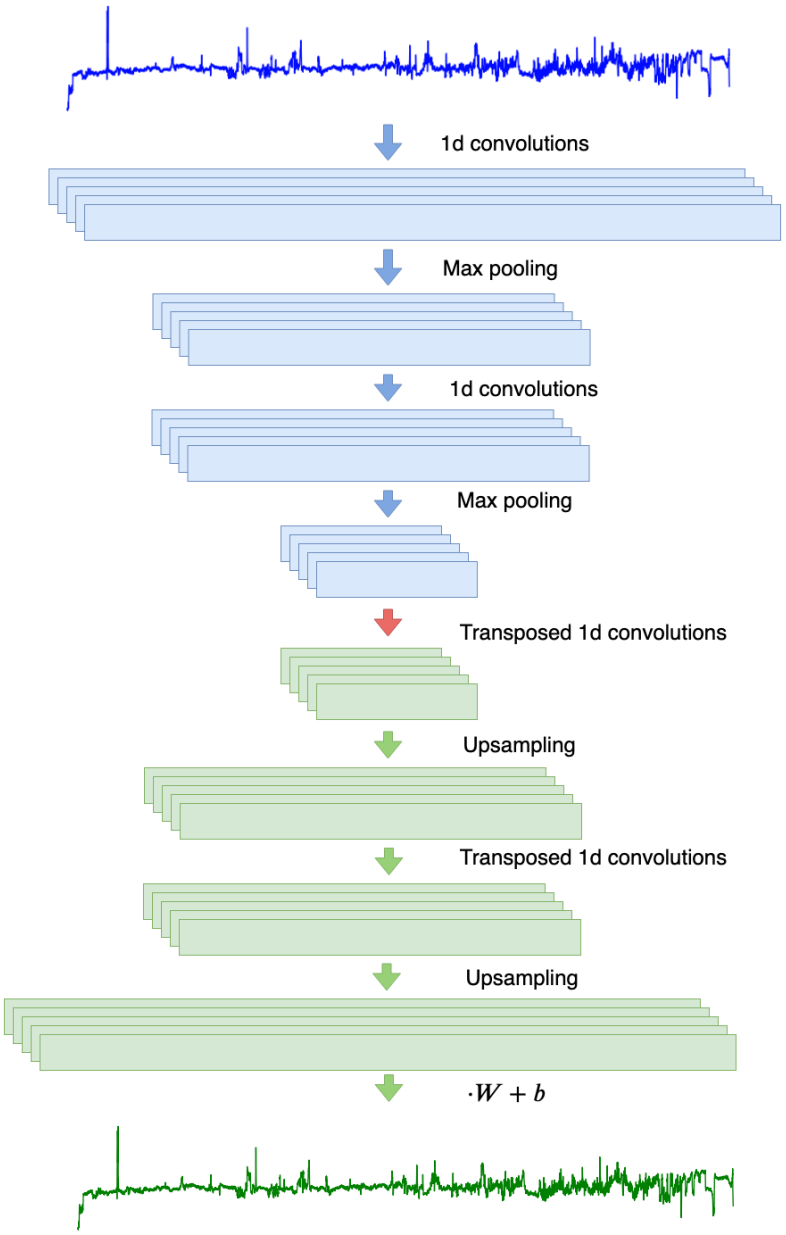


Figure 5.22: Layers of a CAE. The blue parts make up the encoder, and the green parts make up the decoder. Each hidden representation has 5 channels. The max pooling layers reduce the encoding size, while the upsampling layers increase the size. The convolutional and transposed convolutional layers keep the size unchanged. The channels of the final layer are combined by multiplication with weights $\boldsymbol{W}$ and addition of bias b to make the reconstruction of the input sequence.

PCA model that is introduced next, no more effort was put into making the CAE work better. More discussion of the poor performance of the CAE follows in section 5.3.4.

conv	max p.	out channels	channel width	encoding size
		1	39 516	39 516
5,1	6,6	30	6586	197 580
5,1	6,6	30	1098	32 940
5,1	6,6	30	183	5490
5,1	6,6	4	37	148

Table 5.4: CAE encoder setup. The first column presents the width and stride used by convolutional layers, while the 2nd column presents the width and stride used by max pooling layers. The encoding size is given by channel width $\times$ number of out channels.

In the CAE, all biases were initialized to zeroes, and all weights were initialized drawing from a standard normal distribution. The mean squared error was used to calculate the loss, comparing the reconstructed sequence to the original. The Adam optimizer was used with learning rate 0.001.

Principal Component Analysis (PCA)

The implementation of PCA from the `decomposition` module in scikit-learn [Pedregosa et al., 2011] was used for the PCA models. As the amount of features was large, randomized SVD by the method of Halko et al. [2011] was used to find the requested number of components. Calculating the full SVD was neither necessary nor feasible. The PCA model was fit to the training data, which entailed finding the right singular vectors making up the matrix $\boldsymbol{V}_k$. The data was encoded by applying the PCA transformation $\boldsymbol{T}_k = \boldsymbol{X}\boldsymbol{V}_k$. Reconstruction was performed by applying the inverse PCA transformation to the encodings, such that $\boldsymbol{X}_k = \boldsymbol{T}_k\boldsymbol{V}_k^T$. The reconstruction error obtained from comparing the reconstructions to the original sequences was used in the anomaly score calculations.

Anomaly Score

The anomaly likelihood score calculation ran into trouble with the increased size of the reconstructions, due to the huge error vectors. Even with the precautions taken, the anomaly likelihood calculation turned numerically unstable with almost 40000 elements in the reconstruction error vector.

Assuming independent errors, the calculation simplifies to the product of the densities of individual Gaussian distributions per segment. This was also not

useful, since the product of 40000 number all less than one also was numerically unstable. This might have been overcome by calculating individual likelihood scores and summing them, but the assumed independence in this approach is not attractive. The interactions between the terms of the error vector are believed to be instrumental in detecting anomalies. An anomaly of assumed local character will probably be characterized by one or a few sections of the cable having abnormally high individual errors compared to the other sections. The interactions between the terms are captured by the covariances in the multivariate Gaussian distribution, used in the original anomaly likelihood calculation in equation 5.2.

A simplification was engineered that looks at each likelihood score in context of all the other scores. The proposed scheme first places univariate Gaussian distributions upon all of the segments. From the validation error vector, the means of every individual anomaly score $\boldsymbol{\mu}$ is calculated. Also the individual variances $\boldsymbol{\sigma}^2$ are calculated. From these the individual anomaly likelihood scores $\boldsymbol{s}^{(\tau)}$ for time step τ can be calculated, as shown in equation 5.6. The proposed anomaly score for the reconstructed sequence is the maximum individual anomaly likelihood score divided by the mean anomaly likelihood score among all segments, as shown in equation 5.7.

$$\boldsymbol{s}^{(\tau)} = \frac{(\boldsymbol{e}^{(\tau)} - \boldsymbol{\mu})^2}{\boldsymbol{\sigma}^2} \tag{5.6}$$

$$a^{(\tau)} = \frac{\max(\boldsymbol{s}^{(\tau)})}{\text{mean}(\boldsymbol{s}^{(\tau)})} \tag{5.7}$$

With this approach, any one individual anomaly likelihood score that sticks out as being significantly more unlikely than the rest in a moment in time, will lead to a high anomaly score.

A disadvantage of this new score is that it is not as directly interpretable as the Gaussian based anomaly likelihood score. It also strongly assumes that anomalies will be of a local character, and will fail at marking very unlikely sequences if all the individual anomaly likelihood scores are unlikely. An advantage is that it directly admits locating the anomaly in the cable. The anomaly is believed to be in the segment with the maximum individual anomaly likelihood score.

5.3.3 Experimental Results

The anomaly detection results for the spatial models are presented in table 5.5 in the form of average AUC scores. The results are averaged over the faults constructed for each of the three segments, and over the five test folds. The table also presents the mean reconstruction error on the test set, averaged over the five

test folds. To make the result table easier to review, the best result in each column is boldfaced. The mean ROC curves can be seen in figure 5.23.

Model	Encoding size	Instant faults	Gradual faults	Test R. MSE
CAE	4650	0.7476	0.6152	0.002455
PCA	10	0.9935	0.8238	0.0008847
PCA	25	0.9997	0.8898	0.0003754
PCA	50	0.9997	0.9017	0.0001622
PCA	100	1.0000	0.9177	8.442e-05
PCA	500	**1.0000**	0.9251	4.679e-05
PCA	1000	**1.0000**	**0.9271**	4.271e-05
PCA	2000	**1.0000**	0.9257	3.836e-05
PCA	4000	0.9999	0.9210	**3.296e-05**

Table 5.5: Mean AUC scores and reconstruction MSE on the test set for the spatial models

For both types of faults, it is clear both from the table and the ROC plots that all PCA models are significantly better than the convolutional autoencoder. Statistical analysis using the Wilcoxon signed-rank test confirms this, and the p-values of the analysis can be seen in tables A.13 and A.11 in the appendix. The PCA models with 500, 1000 and 2000 components got perfect scores for the instant faults. They are even better at catching the instant faults than the temporal models. For the gradual faults, the PCA model with 1000 components is superior among the models tested so far. It is statistically better than all the other spatial models, except for the PCA model with 500 components. The P-values of this analysis can be seen in the appendix in table A.14. The largest tested PCA model with 4000 components has the smallest reconstruction error on the test set, while the CAE has the largest.

The experiments concerning the spatial dimension have found the best anomaly detection model so far, which is one that might be difficult to beat. Now, the results of this part of experiments are discussed. In particular the disappointing results of the autoencoder in comparison to PCA are analysed. Also the encodings of both PCA and CAE are inspected to see how well they represent whole sequences of DTS values.

5.3.4 Discussion

This section discusses the results of the spatial models.

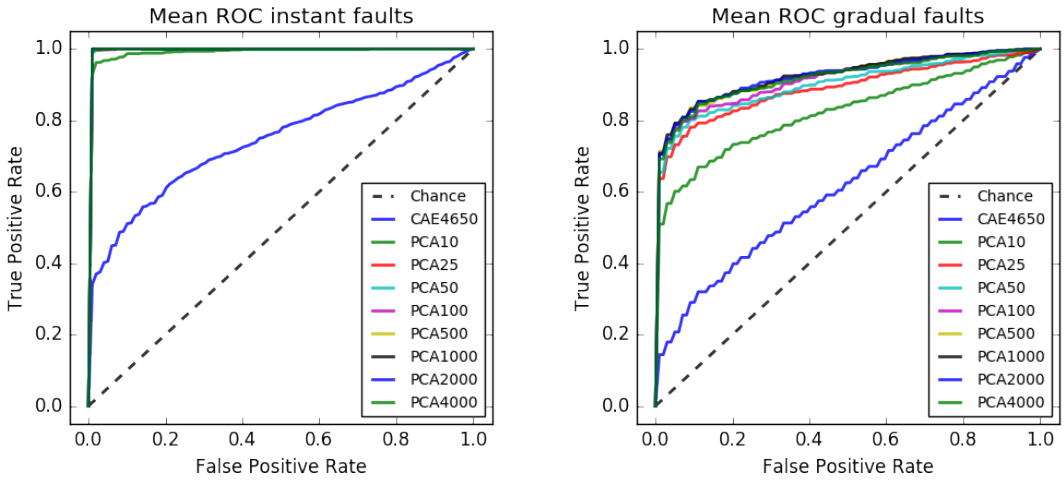


Figure 5.23: Mean ROC for the spatial models. Instant faults are on the left and gradual faults are on the right.

Convolutional Autoencoder

Figure 5.24 shows the anomaly scores for the CAE anomaly detection model applied to the instant and gradual faults of segment 6285 for test fold 2. The red dots are the scores of anomalies, while the blue dots at the same id are the scores of the corresponding normal samples. The plots show that the CAE poorly divides the anomalous and normal samples. Many of the dots appear purple, which happens where normal sample and corresponding constructed fault get the exact same score. This is due to another segment than the anomaly getting the largest individual anomaly score, and thus the overall score for the sample is not affected by the introduced anomaly. This could mean that anomaly score calculation was sub-optimal. It does however work very well for PCA, so the calculation of the score is not the problem.

The CAE got the highest reconstruction error among the tested models, and figure 5.25 shows reconstructions produced by the CAE. The representation learned by the network results in a reconstruction that is a smoothed version of the original sequence. This is not good enough for anomaly detection. The poor reconstruction makes the extra error imposed by the anomaly not stand out as significant. The individual anomaly likelihoods of each segment for the plotted sequences can be seen in figure 5.26. For the plotted fault, the score does get higher for the induced anomaly, but the difference is not very significant. Overall the model does not successfully separate the normal and anomalous samples.

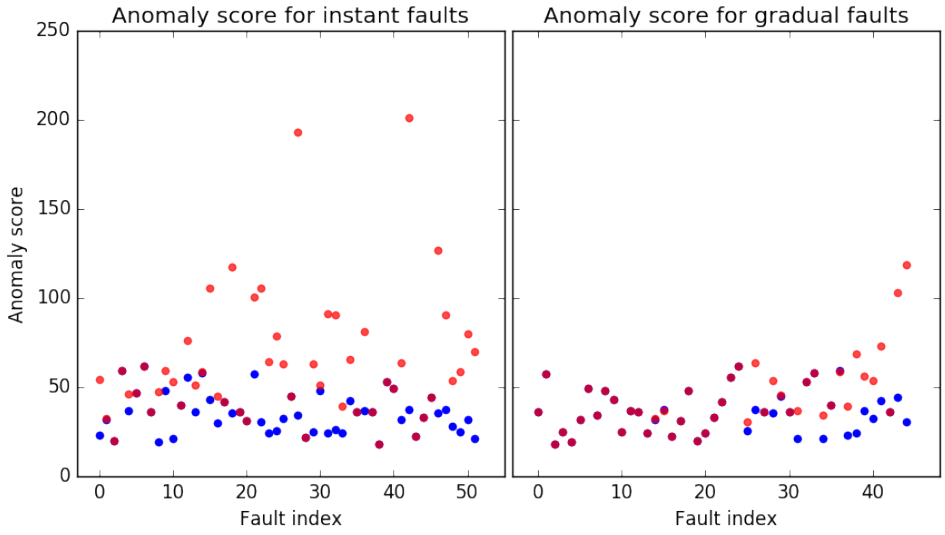


Figure 5.24: Anomaly scores for convolutional autoencoder reconstruction model for instant and gradual faults of segment 6285, test fold 2. The blue dots represent normal samples and the red dots at the same index the corresponding constructed fault.

It is believed that a spatial autoencoder model can achieve results at the same level as the PCA model, with a more suited architecture. Deep learning models should be able to learn reconstructions at least as good as PCA when given enough parameters to tune[1]. Not being constrained to only use linear relationships could even give them the edge to outdo PCA.

Possible improvements to the current implementation include making the encoder and decoder share weights. Also the upsampling operation used might not be optimal, as the linear interpolation leads to large values bleeding into sections were the values are smaller, as can be seen in figure 5.27. Plain broadcasting might be better. Twin objectives of detecting anomalies and producing small encoding were chased when constructing the model. With anomaly detection as the end goal, it might be preferable to focus solely on anomaly detection and thus allow larger encodings.

Principal Component Analysis (PCA)

Figure 5.28 shows the anomaly scores of the PCA anomaly detection model with 500 components applied to instant and gradual faults in segment 6285 for test

[1]A neural network with a single hidden layer with n nodes and linear activations can theoretically learn the top n principal components by gradient descent.

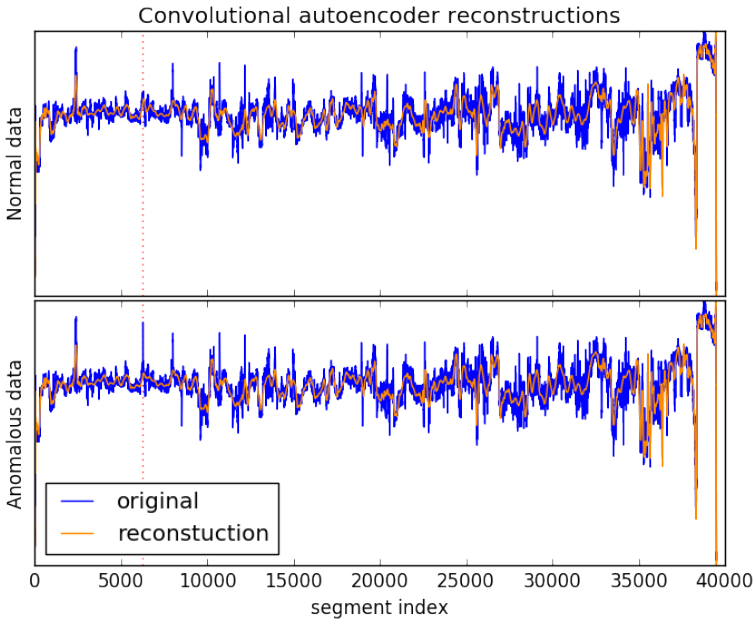


Figure 5.25: CAE reconstruction. The blue graphs are the original sequence of DTS values. The reconstructions are drawn on top. The top plot shows a normal sequence. The bottom plot has an error marked by the dotted red line.

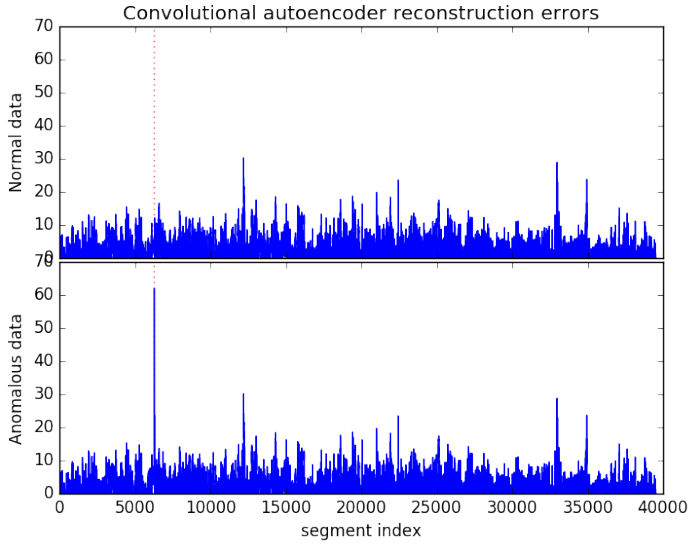


Figure 5.26: CAE reconstruction error for a normal and an anomalous time step

```
[3.0, 4.0, 5.0, 5.0, 6.0, 9.0, 9.0, 8.0, 8.0, 1.0, 3.0, 1.0]
                    [5.0, 9.0, 9.0, 3.0]
[5.0, 6.1, 7.2, 8.3, 9.0, 9.0, 9.0, 9.0, 7.9, 6.3, 4.6, 3.0]
```

Figure 5.27: Example of the max pooling and upsampling operations. The first list is the original list. The second line is the result of applying the max pooling operation to the original list. The last line is the result of applying the upsampling operation to the list in the second line.

fold 2. A very clear decision boundary between the anomalous points and normal points for the instant faults can be seen in the left plot. The gradual faults are plotted to the right. Here a very clear trend can be seen of an increase in the anomaly score of the faults, with the rising temperature increase moving towards the right. For these scores, a line can be drawn that separates the normal data and the faults from day 17(id 9). Here the temperature increase is 2%.

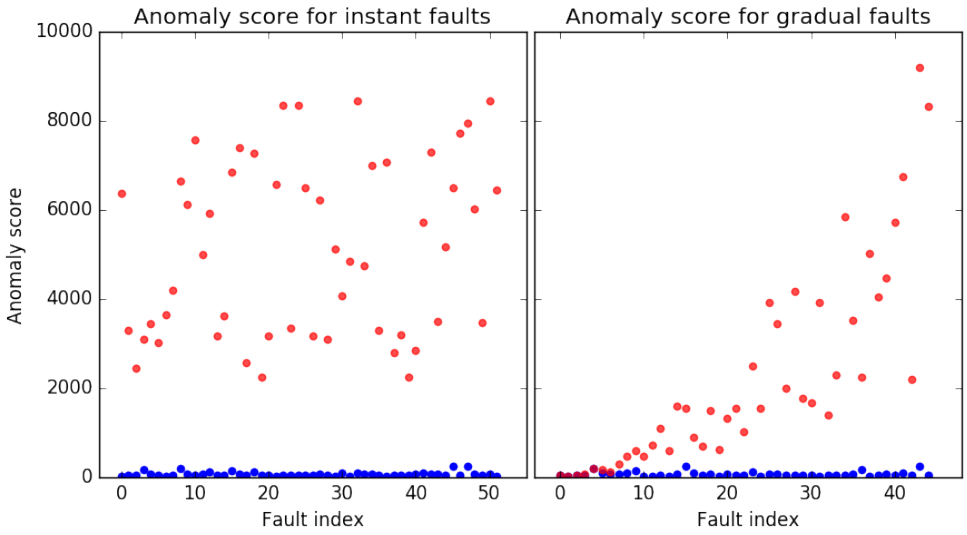


Figure 5.28: Anomaly scores for the PCA reconstruction model with 500 components for isntant and gradual faults.

Figure 5.29 shows how successful a PCA model with 500 components is at reconstructing the normal sequence in the top graph. In the bottom graph the induced error of a 10% increase clearly stands out as not being reconstructed. Comparing the reconstructions produced by PCA to the ones made by the CAE in figure 5.25, it is clear that PCA is able to capture a lot more of the variation along the cable. It is a much better representation of the DTS values, at a smaller

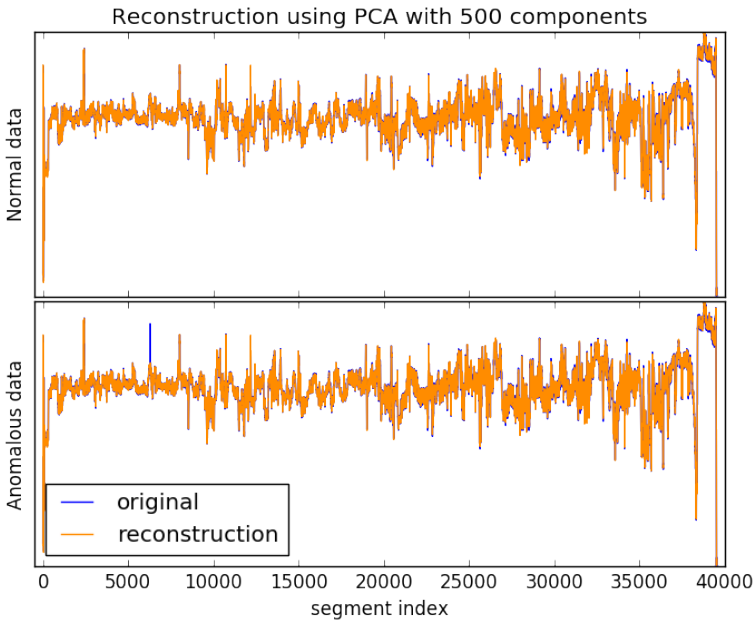


Figure 5.29: Reconstructions produced by the PCA model with 500 components. The top plot shows the normal data, while the bottom shows how the reconstruction of the instant faults in unable to reconstruct the segments with the fault.

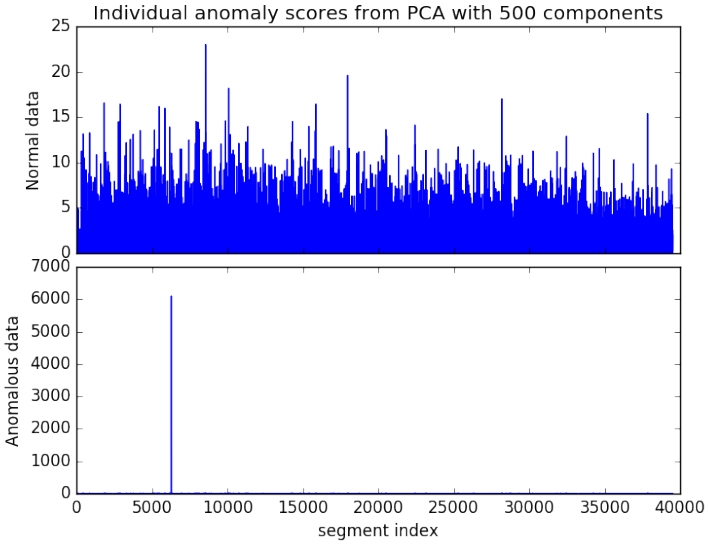


Figure 5.30: Individual anomaly scores for the PCA model with 500 components. The top plot shows the errors of the normal data, while the bottom shows how the errors for the corresponding fault. Note the different scales of the y-axes.

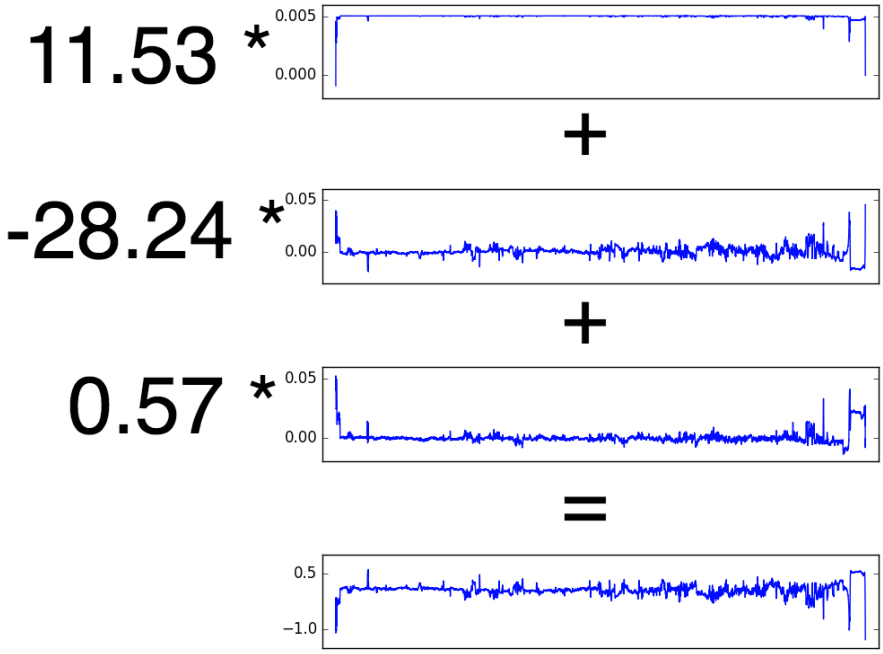


Figure 5.31: Three principal components combined to a reconstruction of the export cable temperatures.

size. The individual anomaly scores of the segments in figure 5.30 show that the model very successfully distinguishes the anomalous part of the cable.

Figure 5.31 shows three first components learned by PCA, and how linearly combining them results in a reconstruction of the cable. Adding more principal components make the reconstructions more accurate, as can be seen in table 5.6. The continued decrease in validation and test MSE with additional components shows that PCA is not overfitting to the training data.

The PCA model with the best anomaly detection results uses encodings of size 1000, which is only 2.53% of the original length of the sequence of DTS values. The anomaly detection performance monotonically increases with the number of component up to 1000. Further increasing to 2000 components leads to a decrease in the performance for gradual faults. It was expected that the performance would eventually start decreasing with further increases in the number of components. When given enough components, PCA will eventually be able to reproduce all the nuances in the cable, also the anomalous parts. This will lead to low reconstruction errors for anomalous behavior, and thus the model will not be able to use the reconstruction errors to distinguish normal and anomalous behaviour. Using too few components leads to poor reconstructions also for normal behaviour. This again leads to the model not being able to distinguish normal and anomalous behaviour,

Components	train MSE	val MSE	test MSE
10	0.0008438	0.0008864	0.0008847
25	0.0003456	0.000376	0.0003754
50	0.0001451	0.0001623	0.0001622
100	7.515e-05	8.436e-05	8.442e-05
500	4.245e-05	4.682e-05	4.679e-05
1000	3.685e-05	4.274e-05	4.271e-05
2000	2.958e-05	3.839e-05	3.836e-05
4000	2.043e-05	3.298e-05	3.296e-05

Table 5.6: Reconstruction errors of the PCA models

this time because both will have high reconstruction errors. Table 5.7 shows the amount of variance in the data explained by different numbers of components. The first component explains over 98% of the variance, but the nuances explained by the first 1000 components are needed to increase the anomaly detection abilities. Further increasing to 2000 leads to 0.005% more of the variance being explained, and this seems to be past the tipping point of being too much.

Components	Ratio
1	98.9288%
10	99.9209%
100	99.9930%
1000	99.9965%
2000	99.9970%

Table 5.7: Ratio of the variance in the DTS data explained by different numbers of principal components

It is clear that with the current implementations, PCA is the choice of the spatial model to move forwards with. It has more or less solved detecting instant faults, and also detects the gradually developing faults at an early stage. It produces very good reconstructions of the sequence of DTS values from an efficient encoding. It also has the added benefit of being fast to fit. The next section tries to combine the spatial and temporal dimension, to see if the PCA model can be enhanced to find the gradually developing faults at an even earlier stage.

5.4 Combined Experiments

Now that the spatial and temporal models have been tested and discussed separately, it is time to combine them. The inputs to the combined models vary, but have in common that they include multiple timesteps of observations of all the 39516 DTS values. Additionally, some of the models also use observations of the electric current, sea temperature and time between samples. This way the amount of data is even larger than it was for the spatial models. The anomaly score calculation proposed in equation 5.7 is used for the combined models as well.

Experiments are executed to see if the PCA model can be enhanced to find the gradual faults at an even earlier stage. By analysing the spatial distribution over time, the model might pick up on the fact that the differences between neighbouring segments are growing, and this might lead to earlier detection.

5.4.1 Experimental Plan

Temporal information is introduced to the most promising model for anomaly detection evaluated so far, namely the PCA model. First different input formats are tried with the PCA model, so that multiple timesteps are part of the reconstruction. Also a promising temporal model is merged with the PCA model, such that the encodings produced by PCA are fed to a prediction model. The next section present the details of the setup of the tested combined models.

5.4.2 Experimental Setup and Architectures

A simple approach that introduces the temporal dimension into PCA is to concatenate multiple timesteps to use as input to the PCA setup described in section 5.3.2. This was implemented and evaluated with concatenations of 3 consecutive timesteps, resulting in inputs as can be seen in figure 5.32. A further extension to this approach also concatenates the control variables so that they also have to be reconstructed.

A more involved extension combined prediction and PCA, the best models of the temporal and spatial experiments. PCA was fit and applied to the DTS data of the training set. The top components were kept and the resulting encodings of reduced dimension were used as inputs and targets to the prediction model. The prediction of the next step was then decoded by the PCA model, and the full prediction error of the whole cable was used to calculate the anomaly likelihood score.

The LSTM prediction model showed some promise among the temporal models, and thus a PCA-LSTM prediction model was implemented. A single-layer LSTM with 1000 units was used. Due to memory constraints only 10 input steps of

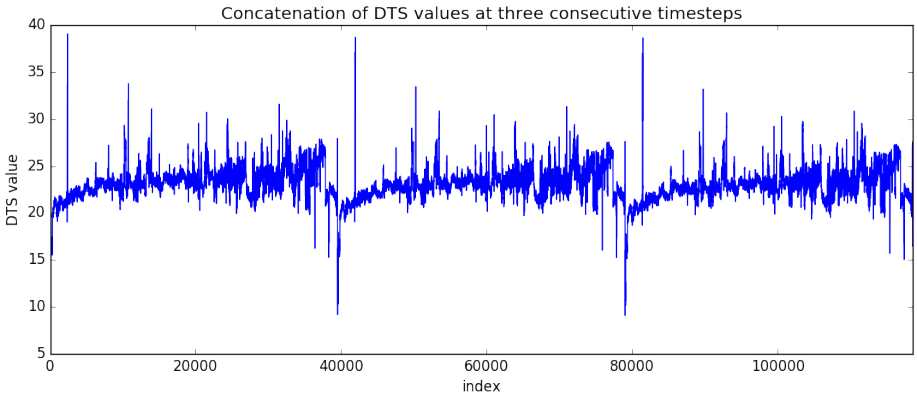


Figure 5.32: The DTS values of three consecutive timesteps concatenated.

encoded DTS values with 100 components, together with the control variables, were used to predict the encoding of the next observed DTS values. As before, the time until the next prediction was concatenated with the last LSTM output. The result of this operation was fed through a fully connected layer with linear activation. The weights of the fully connected layer were initialized by drawing from a standard normal distribution, while the biases were initialized to zeroes. The Adam optimizer was used with a learning rate 0.001, and the loss function was the MSE obtained by comparing the predicted encoding to the target encoding. The MSE was weighted by the variance explained by the associated principal component to ensure the quality of the reconstructions.

Table 5.8 summarizes the dimensions of the inputs, encodings and outputs of the combined models.

Model	Input dim.	Encoding dim.	Output dim.
PCA	3×39156	150	3×39156
PCA c	$3 \times (39156 + 2)$	150	$3 \times (39156 + 2)$
PCA-LSTM	$10 \times (39516 + 3)$	$10 \times (100 + 3)$	$1 \times (39516)$

Table 5.8: Input, encoding and output dimensions of the combined models. For the input and output dimensions, the first number described the number of timesteps, and the second number the dimension of each time step. +2 means that the electric current and sea temperature is included. +3 means that the electric current, sea temperature and time since last observation is included

5.4.3 Experimental Results

The results of the combined experiments are presented in table 5.9. As before, the models have been evaluated on the faults constructed for three different segments in five folds, and the reported results are the averaged AUC-scores.

Model	Instant faults	Gradual faults
PCA	0.9999	0.9064
PCA c	0.9999	0.9007
PCA-LSTM	0.9973	0.8399

Table 5.9: Mean AUCs for the combined models.

The best among the tested combined models was the PCA model reconstructing three consecutive time steps. None of the models are however improvements from the spatial models of comparable encoding size, and they are all statistically worse than the PCA model with 1000 components at the gradual faults. P-values from the statistical analysis confirming this can be found in table A.16 in the appendix. The next section further discusses the results.

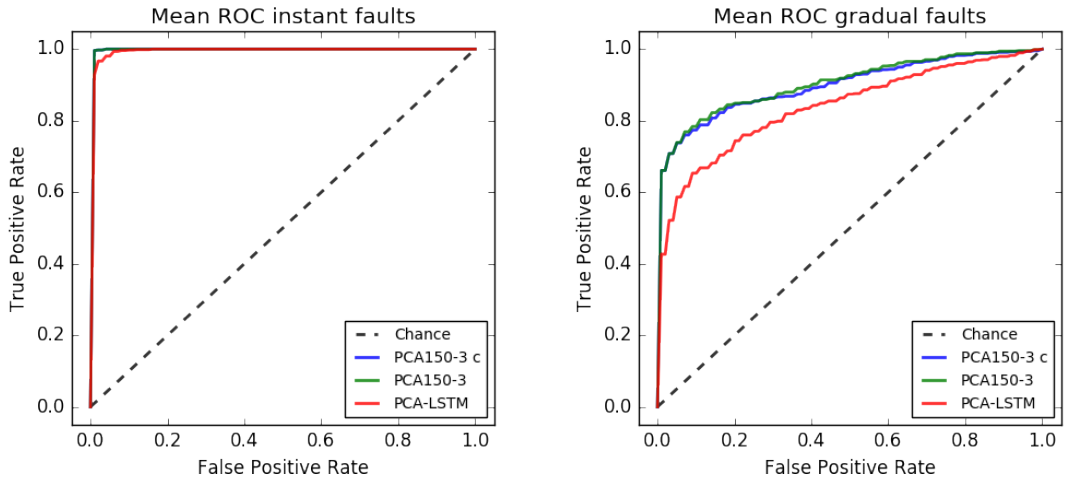


Figure 5.33: Mean ROC for the combined models. Instant faults are on the left, and gradual faults on the right.

5.4.4 Discussion

As speculated, the best spatial model proved difficult to beat. None of the tested combined models achieved the same performance as the PCA model of comparable

Model	Encoding size	Instant faults	Gradual faults
PCA	50	0.9997	0.9017
PCA	100	1.0000	0.9177

Table 5.10: Mean AUC for a subset of the spatial models for comparison for the combined models.

encoding size.

The simplest attempt at introducing the temporal dimension into PCA did not succeed. Including multiple time steps to be reconstructed by the PCA model gave worse performance than only including one step. This can be seen by comparing the results of the extended PCA models to the results of the PCA models of comparable size in table 5.10, that was copied from the spatial results. The single step PCA model with encodings of size 50 was beaten by the 150 component multi-step PCA model for all fault types, but not by much. The single step PCA model with 100 components beats the multiple step PCA with 150 components on both fault types. There seems to be more to gain by getting the encoding size correct for reconstructing a single time step, rather than trying to reconstruct multiple steps.

Including the control variables in the pure PCA model reconstructing multiple steps gave worse results than not including them. More information had to be reconstructed from the same sized encoding, which might have hurt reconstruction. The relationship of the temperature through the cable and the electric current is believed to be of a lagged character, and this might explain why including the current to be reconstructed does not enhance performance. The electric current is probably more vital for prediction models than for models purely based on reconstruction.

The PCA-LSTM model was not successful at beating the pure PCA model with the same encoding size. The points made in the discussion of the temporal models hold here as well. The prediction problem is difficult when so much can happen to the electric current in between time steps, and might thus not be appropriate for anomaly detection in the export cables. Also not many time steps of history were provided to the model, so it did not have many examples from which to extract information about the current physical properties of the cable surroundings.

The memory constraints encountered when combining PCA and LSTM can be overcome, so that more than 10 time steps can be allowed in the analysis. The PCA could be calculated from the training data and applied to every time step of both the training and validation data. The full DTS sequences would then not need to be kept in memory while training the model, which is though to be the reason for the memory issues experienced.

Even though the linear regression model was crowned as the most suited temporal model to expand, it has not been part of the experiments of the combined models. This is a consequence of first evaluating the models based on a threshold set to the 0.999 quantile of the anomaly scores, which lead to marginally better results for the LSTM prediction model than for the linear prediction model. Changing the evaluation to look across thresholds judges linear regression as better, and the prior results might be a result of the chosen thresholds being more optimal for the LSTM model.

It is however believed that the poor predictability of the DTS values might make reconstruction more promising than prediction, when combining the temporal and spatial dimensions. The PCA model reconstructing 3 time steps did get very good results, just not as good as only using the spatial dimension. After the CAE has been improved, it can be extended to use two dimensional convolutional layers, such that it can convolve over both time and space. Thus it was not prioritized to combine PCA and linear regression for predicting the next time step of DTS values.

5.5 Conclusion

This chapter ends with an attempt to answer the second research question, again repeated for convenience:

Research question 2 *Does exploiting relationships in both the temporal and spatial dimensions of the DTS-data give more robust anomaly detection models than only regarding one of the dimensions?*

From the experiments of this chapter, the spatial dimension is believed to carry the most important information for detecting anomalies. None of the combined models succeeded in beating the best spatial model. It is still believed that monitoring the spatial distribution over time might lead to even earlier detection of the gradual faults than only regarding the spatial information in a single time step. This way, a slight drift of a few segments from their neighbours might be captured at an earlier stage. More experiments are however needed to find models that better analyse the temporal evolution of all the DTS values.

In the introduction, the physical laws governing the relationships of the cable and its surrounding variables were emphasised. It is believed that the electric current is the strongest driver of the temperature developments in the export cable. It is thus unexpected that the best anomaly detection models completely disregard the electric current. A single time step of DTS values is made up of 39516 individual values. From the results of the experiments, it is believed that these values are capable of capturing a lot of information about both the state of

the surrounding variables, and the recent history of the electric current. Explicit information about the electric current or sea temperature is thus not necessary for the model. The electric current is still believed to be the biggest driver of the temperature evolution. But also the history of local sea temperatures, the temperature and thermodynamic properties of the distributed surrounding of the cable, the sea current and probably other unknown variables affect the development of the temperatures. A lot of this information is believed to be implicitly captured in the DTS value distribution in the cable. Thus the cable observations themselves give the best summary of the history of its environment, and a single time step of the DTS data might still prove to give unbeatable anomaly detection results, even after further experiments.

Chapter 6

Conclusion and Future Work

This chapter concludes this thesis with a discussion and evaluation of the obtained results. Further the contributions of the thesis are summarized, and future work is proposed.

6.1 Discussion and Evaluation

The tested models have only been evaluated on synthetic faults, as these are the only faults that are available. The faults were only constructed for three random segments. It might be that some of the other cable segments are more difficult to find faults in, and some segments have been observed to experience more variation than others. Hopefully, the models will still be able to differentiate what variation is within normal behavior for each segment. This has however not been evaluated, and thus the quantitative results of the experiments with regard to the AUC scores should not be taken to literally. They are believed to give a potentially inflated estimate of the expected performance of the models in a deployed setting. The qualitative results are believed to be more accurate, such that the best performing PCA model is believed to really be the best model among the ones tested.

It might also be that the assumed characteristics of the faults correspond poorly to the characteristics of future failures in the export cables. The faults are assumed to be of a local character, and this has influenced the proposed calculation of the anomaly score for the spatial models. As a result, the models will not detect global faults with the current anomaly score. This is not a problem if other measures are in place to detect those. A measure to detect global faults can even be devised from the same underlying anomaly detection model. The local anomaly score, calculated as the maximum individual score divided by the mean, can be supported by a global anomaly score. This can also come from the individual anomaly likelihood scores of each segment, and can be the sum or the mean of the individual scores.

All tested models for anomaly detection are based on the error of comparing an expected result to what is observed. This is obvious for prediction models, as they compare a predicted future value to the real observation. Also the reconstruction based models do a comparison of what is expected to what is observed. The reconstructed sequence can be seen as what the sequence was expected to look like, given that the cable had behaved normally. That the anomaly detection models depend on comparing what is observed to what is expected contribute to making them interpretable, as the expected value can be compared to the observed. This can be visualised and inspected to see what gave rise to an anomaly warning. This is a strength of the models, as it gives the operators a chance to inspect the deviations. It also facilitates locating the faults, without which the models would be a lot less useful.

The cable might experience more extreme values than what have yet been observed. Exposure to higher sea temperatures or longer stretches of continuous power production at full capacity can lead to higher temperatures than have yet been observed in the cable. This can lead to the models having to extrapolate. The first series of experiments showed that machine learning models are sensitive to input data with different distributions than the data used for training, and the negative impacts of extrapolation. Under these conditions, the models cannot be expected to work well. The current models have no way of warning when this happens. Frequent retraining will contribute towards preventing extrapolation, but there should still be a measure in place to warn of extrapolation if it should happen. The global anomaly score could potentially warn of extrapolation, if it leads to large reconstruction errors over the whole sequence.

The division of the data into blocks with two days of consecutive data can be argued to imply frequent retraining of the models. In the experiments, the models were never evaluated on data from time periods more than a few days away from the data used for training. This prevented extrapolation in the controlled training and evaluation setting of the experiments. The small block size was also convenient for constructing samples of normal and anomalous data for evaluation. It is believed that the models realistically can be retrained as frequently as every second day, although it might suffice to retrain once or twice a week. The training of the PCA model in particular is not very resource intensive, and can easily be automated.

Frequent retraining might however have negative consequences for the detection of gradually developing faults. If a model is retrained frequently and a fault is gradually developing, then the model with be trained with partly anomalous data. The gradual drift will however only constitute a small proportion of the training data when a lot of data is available, which could limit the impact of including the anomalous data. This situation has not been part of the experiments, so the extent of the problem is not known. It is hypothesized that including the drifting

data might lead to slightly slower detection.

A precaution to avoid this issue is to deploy multiple parallel anomaly detection models trained on different temporal subsets of the data. A recent model can be trained with all data to guard against hurtful extrapolation. An earlier model can be trained with data that does not include the last chronological proportion. If a gradual fault is developing, the earlier model will not be trained on the tail of the developing fault, and should thus have better chances of catching it. Any potential anomalies detected only by the earlier model should be inspected. If the high anomaly score is caused by the model having to extrapolate, and the recent model does not regard the behaviour as anomalous, then the cable is probably in order. If the high anomaly score from the earlier model does not seem to be caused by extrapolation, it might have caught a gradually developing fault that has been absorbed as normal behaviour by the recent model. The number of parallel models can be more than two, such that for example one was trained with all data, one excluding the past week and one excluding the past month.

The anomaly detection experiments evaluated the models across thresholds. When deploying a model, a threshold must be set to decide the cutoff between normal and anomalous behaviour. Setting this threshold must be done with attention to the trade-off of having fewer false negatives versus having fewer false positives. For assets of high importance, minimizing the number of false negatives will often be prioritized. This will be at the cost of suffering more false positives. In a deployed setting, the models will meet an overweight of false samples, as most of the behaviour of the system is normal. This motivates trying to constrain the number of false positives, as too many false positives might lead to operators ignoring warnings of real anomalies.

If past faults are available, ROC plots can be used to find acceptable ratios of false positives and false negatives. If faults are not available, as is the case for the export cables, a threshold can be set based only on the statistics of the anomaly scores of normal samples. For example the maximum observed anomaly score, or a high quantile such as 0.999 can be used to set a threshold. This allows somewhat controlling the ratio of false positives, and is not at risk of overfitting to assumed fault characteristics. It does however not offer any control of the rate of false negatives. Alternatively, synthetic faults can be constructed and used for setting the threshold. This is more involved. It gives the same control of the false positive rate, and offers an estimate of the false negative rate. It might however lead to overconfidence, or overfitting to the assumed characteristics of the faults that guide the construction of the synthetic faults. This might in turn lead to faults with different characteristics going undetected.

The PCA model only regarding the sequence of DTS values from a single time step has the best results among the models evaluated in this thesis. It is tuned

by a single parameter, which is the number of principal components to use in the intermediate encoding. Multiple numbers were tried in the experiments, and the best configuration used 1000 components. Choosing this number in a totally unsupervised setting is challenging. Optimizing the reconstruction error does not help, as the reconstructions must be good enough to reconstruct normal behaviour, but not too good as then also the anomalous parts can be reconstructed. Finding the tipping point requires testing different configurations for anomaly detection. Table 5.5 shows a relatively small difference in anomaly detection performance between the PCA models with 100 to 4000 components, so there seems to be a large space of reasonable configurations. From table 5.5, these models all have reconstruction MSE on the test set to the order of $1e-05$. Thus if a general rule were to be made from the experiments conducted, the reconstruction error should be in the order of $1e-05$ to be good enough. A more involved method for choosing the number of components is to make samples with small changes to a segment, and see what number of components best detect the fault.

A similar challenge presents itself when constructing deep learning models to use for anomaly detection. The approach of modelling normal behaviour to use deviations to detect anomalies does not entail explicitly training the deep learning models for anomaly detection. Rather, the models are trained with another objective, being it mean squared prediction error or reconstruction error. This leads to the same challenge as for the PCA model, namely selecting the most appropriate model if no faults have occurred. It might not be the model with the lowest reconstruction error that achieves the best anomaly detection results. The temporal anomaly detection experiments showed that the lowest prediction errors do not necessarily correspond to the best anomaly detection results. Model selection can be solved in the same way as proposed in the previous paragraph for the PCA model, by constructing some simple synthetic faults and choosing a model performing well at detecting these.

To close the discussion, the research questions presented in the introduction are again revisited to see how far the research has come in answering them. Finally the research goal is revisited, to see to which extent it has been achieved.

Research question 1 *How are the asynchronous time series of the export cable system best handled in sequential deep learning models?*

It was found that for prediction, the asynchronous time series of the export cable system could safely be handled by resampling to the uneven rate of the DTS observations. When including time information in the input to the prediction model, the unevenness did not have a significantly negative impact on predictions of the DTS value for the next time step.

The first research question proved not be very consequential to the best anomaly detection model, as it completely disregarded the temporal dimension.

How best to handle asynchronous and unevenly sampled time series data for reconstruction has not been given any attention. The reason for this is that prediction models were hypothesised to be more appropriate for the export cables. The DTS values proved not to be as predictable as hypothesized, and the prediction models had very little success at detecting gradually developing faults. The reconstruction based spatial models were superior by far. Reconstructing multiple time steps might further enhance anomaly detection. For this purpose, it might be valuable to experiment with how best to handle asynchronous time series in reconstruction based deep learning models. The experiments of this thesis resampled the time series to be synchronous but unevenly sampled. How this affected the reconstructions is not known, as no other methods were evaluated for comparison.

Research question 2 *Does exploiting relationships in both the temporal and spatial dimensions of the DTS-data give more robust anomaly detection than only regarding one of the dimensions?*

The second series of experiments found the spatial dimension to be the most successful at anomaly detection in the export cable. A lot of information about the surrounding environment of the cable and the recent history of the electric current is believed to be implicitly captured in the temperature distribution. It is believed that further improving a spatial reconstruction model, or expanding it to also include temporal information, has the most promise for further improvement.

Goal *Find the best deep learning approach to anomaly detection in the submarine export cables.*

This work cannot claim to have found the best deep learning approach to anomaly detection in the export cables. The best model shows great promise at being able to detect gradually developing faults at an early stage, but it is not a deep learning model. Constraining the goal to only include deep learning models was a precaution to constrain the scope of the research, and finding a good solution that does not use deep learning is not a defeat. On the contrary, the PCA model is fast and easy to fit, as compared to the procedure of finding and training a reasonable deep learning architecture. Thus it has many advantages in addition to the observed superior performance.

The PCA model will not be proclaimed to be the best possible approach to anomaly detection in the cables. It is merely the best approach among the ones tested, but it does achieve very promising results.

6.2 Contributions

As many of the reviewed works on anomaly detection, this work builds on the ideas of Malhotra et al. [2015] and Malhotra et al. [2016]. Abundant data of normal behaviour was modelled in order to detect deviations from normality. While Malhotra et al. use this idea only in the temporal dimension, this work has also successfully applied it to the spatial dimension. As in Malhotra et al. [2016], reconstruction error vectors were used to calculate an anomaly score. New is that the reconstructed sequence was a spatial sequence, not a temporal sequence. Also the model used to encode and reconstruct the sequences differ. Malhotra et al. [2016] use LSTMs, while the experiments of this thesis achieved the best results with a shallow PCA model. Finally the anomaly score at the center of the anomaly detection was altered to be usable with higher dimensional inputs.

Araujo et al. [2018] reported that SVD was unsuited for anomaly detection in their DTS system. It was reportedly computationally expensive and better suited to small sections of pipeline without much variation. This experience is contrary to the results of this thesis. Using randomized SVD by the method of Halko et al. [2011] to get the top few thousand singular values was efficient for the whole of the export cable. Getting the whole SVD was not efficient, but this is not problematic as it is not needed for reconstruction based anomaly detection.

Araujo et al. [2018] reported using a CAE model that regarded both temporal and spatial dimensions. They do not report on whether it was evaluated against models only regarding the spatial dimension. The experiments of this thesis got poor results using a CAE. More work can be put into making it better, as has been discussed. The poor results obtained do show a weakness of deep learning methods as compared to PCA. They can be more difficult to get to work, as they have a lot more hyperparameters to tune. For PCA, the only hyperparameter is the number of components to keep. For CAE the number of possible architectures is infinite, with different combinations of the number of layers, activation function, convolutional width, stride, width of the max pooling layers etc.

Araujo et al. [2018] only report analysing the data from the DTS system. This work tried to include more information than only DTS data, by also including the electric current and sea temperature. The efforts did not enhance the anomaly detection abilities. It has been speculated that the information in the control variables is implicitly captured within the export cable temperature distribution, so that nothing is gained by explicitly including the variables.

Extending on the work of Bergmeir and Benitez [2012], blocked cross validation was altered to use multiple distributed small blocks rather than large sequential blocks. This makes the approach suitable also for periodical time series when few periods have been observed. This approach should be valuable to other domains where the data has annual periodicity, as a long time has to pass before many

periods are observed.

In summary, the main contributions of this thesis include:

- A proposed complete anomaly detection scheme for the export cables. It is based on using PCA to encode and reconstruct the sequence of DTS values at a single time step. The reconstruction error is used to differentiate normal and anomalous behavior.

- An anomaly score for high-dimensional error vectors when faults can be assumed to be of a local character, as proposed in equation 5.7. It uses individual univariate Gaussian distributions placed on each element of the error vector, and the score is the maximum individual likelihood divided by the mean individual likelihood. This enables using the anomaly detection method of Malhotra et al. [2015] with high-dimensional data.

- A proposed scheme for how to train and evaluate models with time series data with seasonality, using cross validation with small sequential blocks randomly distributed to the different folds. It builds on the work of Bergmeir and Benitez [2012], but better facilitates periodical data when few periods are available.

6.3 Future Work

Throughout the discussion of the anomaly detection models, possible improvements of the current models have been proposed that can constitute future work. Spatial models have been crowned most promising for further improving anomaly detection in the export cables, and should receive the most focus. In particular, more attention can be paid to spatial deep learning models. As discussed, they should be able to learn to reconstruct at least as well as PCA when given enough parameters to adjust. It is believed that the spatial dimension is able to capture the information needed to capture most types of anomalies. Spatial models can also be expanded to include temporal information by reconstructing multiple consecutive time steps. Even though the initial attempts of combining the dimensions were unsuccessful in outperforming the purely spatial models, other combined models may prove to be successful and should be attempted. As a step in the effort of including temporal information into spatial reconstruction models, how to handle the uneven sampling rate of the DTS system in reconstruction models can be more rigorously explored.

Future work can also focus on making the anomaly detection models more interpretable. Having interpretable model output has been voiced as a wish from the wind farm operators. They are more likely to trust the output of the model

if it can explain its reasoning. Being able to show the discrepancy between the observed and expected values is a step towards achieving this. Getting a measure of the uncertainty of the model can improve trustworthiness. This can be done in the form of confidence intervals, that have the benefit of being easily visualized. It would be useful if the model warned when extrapolating and producing uncertain results. An additional global anomaly detection score has been proposed with this aim, but the idea should be matured and tested.

Due to the anomaly-free nature of the available data, the proposed model has only been evaluated on synthetic faults. Rather than focusing on improving the anomaly detection performance on the constructed faults, it might be useful to evaluate the model on data with real anomalies from other domains. DTS systems are used to monitor other types of systems, as the liquid pipelines that were the subject of Araujo et al. [2018] that had experienced a few faults. Anomaly detection based on PCA reconstruction can also be applied to other data with a spatially distributed nature. Randomly blocked cross validation can be tested more rigorously on multiple datasets, to see whether the results obtained on the export cable data are general results.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Araujo, M. S., Spidle, H. A., Siebenaler, S. P., Blaisdell, S. G., and Vickers, D. W. (2018). Application of machine learning to distributed temperature sensing (dts systems. *Proceedings of the 2018 12th International Pipeline Conference, September 24-28, 2018, Calgary, Alberta, Canada.*

Baytas, I. M., Xiao, C., Zhang, X., Wang, F., Jain, A. K., and Zhou, J. (2017). Patient subtyping via time-aware lstm networks. *KDD17, August 13-17, 2017, Halifax, NS, Canada.*

Bergmeir, C. and Benitez, J. M. (2012). On the use of cross-validation for time series predictor evaluation. *Information Sciences*, 191:192–213. `https://www.sciencedirect.com/science/article/pii/S0020025511006773?via%3Dihub`.

Binkowski, M., Marti, G., and Donnat, P. (2018). Autoregressive convolutional neural networks for asynchronous time series. *Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.*

Chauhan, S. and Vig, L. (2015). Anomaly detection in ecg time signals via deep long short-term memory networks. *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA).*

colah.github.io (2019). Understanding lstm networks. `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`. Accessed: 2019-06-02.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. *Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland, 2008.*

Filonov, P., Lavrentyev, A., and Vorontsov, A. (2016). Multivariate industrial time series with cyber-attack simulation: Fault detection using an lstm-based predictive data model. *arXiv:1612.06676.*

Gal, Y. and Ghahramani, Z. (2015). A theoretically grounded application of dropout in recurrent neural networks. *arXiv:1512.05287.* `https://arxiv.org/abs/1512.05287`.

Halko, N., Martinsson, P.-G., and Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 2(53):217–288.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Huang, G.-B., Zhu, Q.-Y., and Siew, C.-K. (2016). Extreme learning machine: Theory and applications. *Neurocomputing*, 70(13):489–501.

Kofod-Petersen, A. (2015). How to do a structured literature review in computer science. `https://www.researchgate.net/publication/265158913_How_to_do_a_Structured_Literature_Review_in_computer_science`.

Liu, R., Meng, G., Yang, B., Sun, C., and Chen, X. (2017). Dislocated time series convolutional neural architecture: An intelligent fault diagnosis approach for electric machine. *IEEE transactions on industrial informatics*, 13(3):1310–1320.

Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., and Shroff, G. (2016). Lstm-based encoder-decoder for multi-sensor anomaly detection. *ICML 2016 Anomaly Detection Workshop, New York, NY, USA, 2016.* arXiv:1607.00148.

Malhotra, P., Vig, L., Shroff, G., and Agarwal, P. (2015). Long short term memory networks for anomaly detection in time series. *ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning.*

Masci, J., Meier, U., Ciresan, D., and Schmidhuber, J. (2011). Stacked convolutional auto-encoders for hierarchical feature extraction. *Honkela T., Duch W., Girolami M., Kaski S. (eds) Artificial Neural Networks and Machine Learning - ICANN 2011. ICANN 2011. Lecture Notes in Computer Science, vol 6791. Springer, Berlin, Heidelberg.*

Neil, D., Pfeiffer, M., and Liu, S.-C. (2016). Phased lstm: Accelerating recurrent network training for long or event-based sequences. *30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.*

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rencher, A. and Christensen (2012). Principal component analysis. In *Methods of Multivariate Analysis*, chapter 12, pages 405–433. Wiley Series in Probability and Statistics.

Russell, S. J. and Norvig, P. (2016). Learning from examples. In *Artificial intelligence - a modern approach, 3rd Edition*, Prentice Hall series in artificial intelligence, chapter 18. Prentice Hall.

scholar.google.com (2019). Google scholar about. `https://scholar.google.com/intl/en/scholar/about.html`. Accessed: 2019-01-31.

Srivastava, N., Mansimov, E., and Salakhutdinov, R. (2015). Unsupervised learning of video representations using lstms. *Proceedings of the 32 nd International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.*

statkraft.no (2019). Green light for dudgeon offshore wind farm. `https://www.statkraft.com/media/press-releases/Press-releases-archive/2014/Green-light-for-Dudgeon-Offshore-Wind-Farm/`. Accessed: 2019-06-02.

Stec, A., Klabjan, D., and Utke, J. (2018). Unified recurrent neural network for many feature types. *arXiv:1809.08717.*

Strang, G. (2016). The singular value decomposition (svd). In *Introduction to linear algebra*, chapter 7, pages 364–400. Wellesley-Cambridge Press.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Ukil, A., Braendle, H., and Krippner, P. (2011). Distributed temperature sensing: Review of technology and applications. *IIE Sensors Journal*, 12(5):885–892.

UN.org (2019). Climate Change - UN Sustainable Development. `https://www.un.org/sustainabledevelopment/climate-change/`. Accessed: 2019-05-27.

vindportalen.no (2019). Offshore vindkraft. `https://www.vindportalen.no/Vindportalen-informasjonssiden-om-vindkraft/Vindkraft/Offshore-vindkraft`. Accessed: 2019-05-27.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83.

Yan, W. and Yu, L. (2015). On accurate and reliable anomaly detection for gas turbine combustors: A deep learning approach. *Annual conference of the prognostics and health management society 2015*.

Zhang, S., Bahrampour, S., Ramakrishnan, N., Schott, L., and Shah, M. (2017). Deep learning on symbolic representations for large-scale heterogeneous time-series event prediction. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Appendices

A.1 Supportive material to the 1st series of experiments

A.1.1 Exploratory experiments for experiments with asynchronous time series data

The goal of the exploratory experiments was to find a reasonable position in the hyperparameter space. For the experiments all hyperparameters but one were frozen, and one varied to find its best configuration at the given settings of the others. While this approach will not result in finding the absolutely optimal parameters, it should be sufficient to find valuable results. The same hyperparameters are used for all of the following experiments, and all experiments will thus be subject to the same possible suboptimality in the hyperparameters.

The data used for the exploratory experiments were the dts values at their raw sampled frequency, with the electric current and sea temperature resampled using linear interpolation to fit. Cross validation with only training and validations sets was used to validate the results. The evaluation in these exploratory experiments was not very strict, but as mentioned the goal was to find reasonable settings, not necessarily the best ones. A quick approach was thus favoured. They will also be optimized for model A, not the others, which might bias towards this model. There is however not enough time to prioritize fitting the hyperparameters to each individual model.

Initial experiments explored the learning rate to use for the Adam optimizer. Learning rates of 0.01 and 0.001 were tested. The number of LSTM units was kept at 32, and the training input consisted of 15 input time steps. The training target was the following DTS reading, and only the last prediction was utilized when updating the weights. The results of these experiments are presented in table A.1. While none of the learning rates were better across all folds, both seem to be a reasonable choice. While 0.001 is the default in Tensorflow, it was chosen to use 0.01 instead. It seemed to give better results, and also resulted in sligtly faster

training.

lr	train	val	time
0.001	0.045	0.056	34.8min
0.01	**0.042**	**0.053**	25.1min

Table A.1: Validation MSE for two different learning rates when using 15 input steps to predict the next DTS reading with a 32 unit LSTM.

Fixing the learning rate, the number of LSTM units was explored and set. 16, 32, 64 and 128 single unit LSTMs were tried, as well as some stacked versions. Again 15 input steps were used to make 1 prediction of the next DTS temperature. Based on the results presented in table A.2, it was decided to go with a 64 unit configuration for the full-scale experiments.

LSTM units	train	val
16	0.044	0.053
32	0.042	0.053
64	0.046	**0.050**
128	0.043	0.051
32, 32	0.044	0.054
64, 64	0.047	0.051
32, 32, 32	**0.041**	0.053

Table A.2: Validation MSEs for different LSTM configurations using 15 steps to predict the next DTS temperature.

Next, how many steps to use to make updates was experimented with. Two different aspect were of interest, the first being how many input steps to include, and the second being how many predictions to produce and use to update the model weights. A larger number of input steps will provide the model with more context, but might not be necessary and will also make for slower training. As for the number of predictions to use to make updates, the models will be evaluated on making only one prediction of the next DTS reading following the inputs. The LSTM does however produce output for each input step that might be used to make a prediction for the next step. It might potentially serve as a regularizing precaution to make use of more than only the last prediction made to update the weights during training, as this could lead to a more robust representation of the internal state and faster training. It is however more challenging to construct comparable training samples for all the different resampling methods when using multiple predictions.

The results of this experiment is presented in table A.3. A 64 unit single layer LSTM was used for the experiments, and the learning rate was 0.01 for the Adam optimizer. Based on the results, it was decided to go with inputs consisting of 50 inputs steps.

Comparing both rows with input steps and 1 and 20 prediction steps respectively motivates the choice of only using the last prediction for updating the weight during training. This also makes for easier construction of comparable training data for the different models to be tested.

I	P	train	val
5	1	0.057	0.059
15	1	0.042	0.051
50	1	**0.039**	0.049
100	1	0.040	**0.048**
50	20	0.039	0.052

Table A.3: Initial experiments of how many input steps to include, and how predictions to use in loss calculation.

Finally, whether to use dropout regularization was experimented with. Different dropout configurations and rates p of keeping a unit intact where tried. The tried configurations include randomly masking (setting to zero) a proportion of the inputs, randomly masking a proportion of the outputs, a combination of masking both inputs and outputs and variation output applied to the LSTM hidden state as described in Gal and Ghahramani [2015]. The masking was only applied during training. During test time, the weights in the layers where dropout was applied were multiplied with p. Also no dropout was tried. The rest of the setup consisted of a 64 unit LSTM cell accepting 50 input steps to make 1 prediction. The raw DTS data was utilized, with electric current and sea temperature resampled to fit.

Dropout config	train	val
No dropout	**0.044**	**0.048**
Input keep 0.5	8.303	8.675
Input keep 0.8	0.677	1.055
Output keep 0.5	0.085	0.073
Output keep 0.8	0.057	0.053
Variational keep 0.5	0.046	0.051
Variational keep 0.8	0.046	0.051

Table A.4: Results for initial dropout experiments. Results are averaged over the five validation folds.

config	train	val
add	0.049	0.051
concat, sigmoid	0.046	0.052
final merged PLSTM	**0.042**	**0.048**

Table A.5: Results of initial experiments on PLSTM configurations. Results are averaged over the five validation folds.

A.1.2 Supplemental experimental results

Supportive material to section 4.3.2.

Model	val 1	val 2	val 3	val 4	val 5
A	0.053	0.046	0.065	0.055	0.055
B	0.064	0.054	0.081	0.070	0.067
C	0.050	0.047	0.071	0.065	0.054
D	0.054	**0.045**	0.072	0.058	0.058
E	0.439	0.191	0.209	0.151	0.125
F	**0.048**	0.046	0.065	**0.052**	**0.051**
G	0.053	0.051	**0.065**	0.058	0.055
LR	0.056	0.063	0.066	0.065	0.067

Table A.6: Validation MSE for different tested models averaged over each test fold for test data.

Model	train 1	train 2	train 3	train 4	train 5
A	0.047	0.042	0.048	0.044	0.044
B	0.049	0.046	0.058	0.052	0.054
C	0.039	0.046	0.050	0.046	**0.040**
D	0.039	**0.037**	0.045	0.045	0.040
E	0.169	0.120	0.141	0.144	0.113
F	**0.038**	0.039	**0.045**	**0.041**	0.041
G	0.046	0.048	0.055	0.047	0.052
LR	0.052	0.052	0.057	0.055	0.059

Table A.7: Training MSE for different tested models averaged over each test fold for test data.

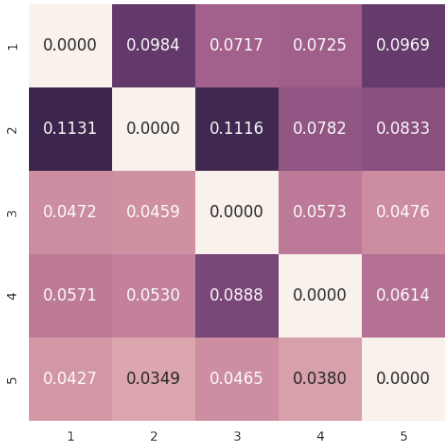


Figure A.1: Full test results for model A by test fold on the y axis and validation fold on the x axis.

Figure A.2: Full test results for model B by test fold on the y axis and validation fold on the x axis.

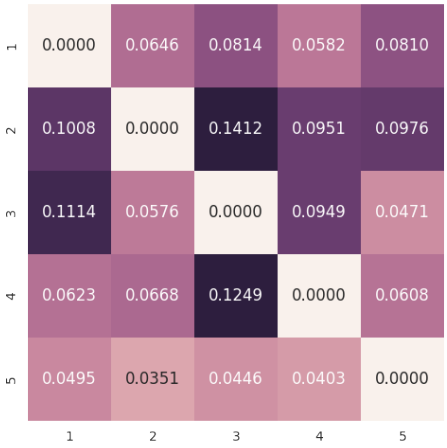


Figure A.3: Full test results for model C by test fold on the y axis and validation fold on the x axis.

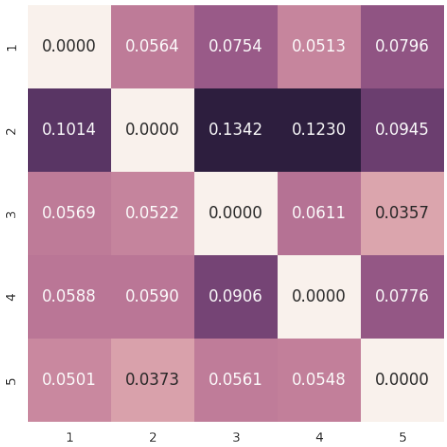


Figure A.4: Full test results for model D by test fold on the y axis and validation fold on the x axis.

Figure A.5: Full test results for model E by test fold on the y axis and validation fold on the x axis.

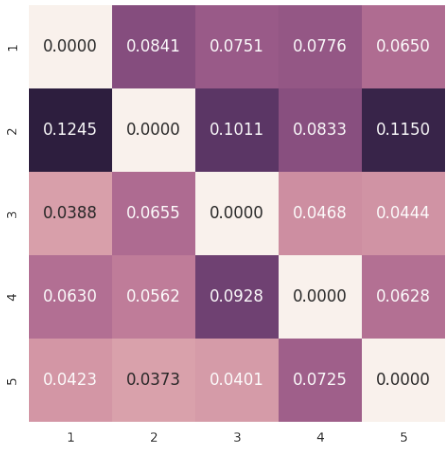


Figure A.6: Full test results for model G by test fold on the y axis and validation fold on the x axis.

A.1.3 Results from rerun with randomly blocked cross validation

Full results from rerun with randomly blocked.

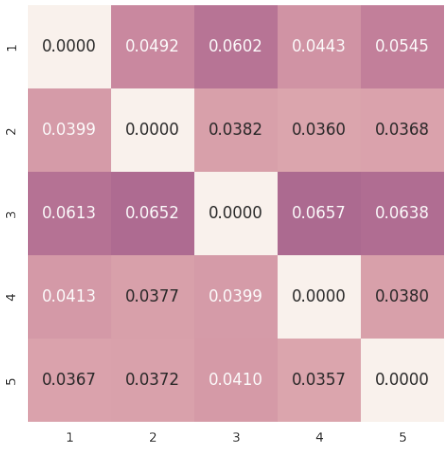


Figure A.7: Second iteration full test results for model A. Test fold is on the y axis and validation fold on the x axis.

A.2 Supportive material to the 2nd series of experiments

Table A.8 presents the results of the initial experiments conducted to decide what state to pass between the encoder and decoder in the LSTM autoencoder of section 5.2.2. The models were trained on data from folds 3, 4 and 5. The 2nd fold was used as the validation data. The 1st fold was set aside for testing, and was not used in the initial experiments. The table reports the MSEs for training and validation data of each tested configuration. Also the train loss is reported. It is different from the train MSE due to using a conditioned LSTM. Thus, the input is different during training than during evaluation on the training set. The experiments reconstructed a sequence length of 50.

state	lr	train MSE	val MSE	train loss
Cell	0.000 100	0.000 959	0.000 800	0.000 218
Hidden	0.000 100	0.000 903	0.000 741	0.000 221
Both	0.000 100	0.000 943	0.000 807	0.000 233
Cell	0.000 500	0.000 232	0.000 239	8.69e-05
Hidden	0.000 500	0.000 222	0.000 231	9.16e-05
Both	0.000 500	0.000 241	0.000 247	9.45e-05
Cell	0.001 000	0.000 132	0.000 144	5.11e-05
Hidden	0.001 000	0.000 122	**0.000133**	4.62e-05
Both	0.001 000	0.000 146	0.000 165	5.81e-05
Hidden	0.010 000	**0.000115**	0.000 146	4.62e-05

Table A.8: Results of initial experiments of what state to transfer between LSTM encoder and decoder.

Failure Prediction for Offshore Wind Turbines
Distributed Temperature Sensing (DTS) system

Failure Scenarios: Characteristics

1 Introduction

This memo describes and quantifies the failure scenarios that need to be detectable on the DTS system at the Dudgeon wind farm. These failure scenarios are based on engineering principles and are the output of discussions between the electrical engineering discipline (OEX TIOP) and analysts in OEX DPA.

To-date (Q1 2019) no failures have been experienced on the system. However, within the data, the algorithm being developed requires failure patterns to learn from. Therefore, it is necessary to create synthetic failures based on the characteristics provided here.

2 Failure Types

2.1 Absolute Temperature Limits

Absolute cable temperature limits can be established from the cable's physical properties. It is **not** necessary for the algorithm to detect or react to these absolute limits as there are already audible warnings and alarms set in the SCADA system within the control room. For reference, the limits are:

- Warning: $85^{O}C$.
- Alarm: $90^{O}C$.

Source: C177-DOW-E-RA-0001 "Cable Conductor Temperatures & Manual Response"

2.2 Instant Failure (step-change)

During an instant failure (or near-instant failure) the cable's physical properties will change rapidly. The DTS systems samples the temperature every c.17 minutes. Therefore, within an instant failure the temperature rise is expected to be evident within one time-step

It is likely that an instant failure will affect the temperature of the cable section over all current ranges since the physical damage is not dependent on temperature. A realistic minimum detectable temperature change for the worst-affect segment is +10%, as

compared to the normal temperature. Due to temperature dissipation, the affected cable segments are likely to experience a normally distributed temperature rise affecting in the region of +/- 4m of cable (with a peak of +10%).

2.3 Gradual Failure (degradation)

If the cable is degrading physically, if the cable is gradually becoming exposed or if a cable free-span is growing, it is expected that a gradual temperature change would be detected by the DTS system.

Due to the potential underlying root causes, the timeframe for this deterioration is difficult to predict. For the purposes of this analysis, a period of 3 months is deemed prudent to detect.

All other characteristics remain the same as the instant failure scenario (e.g. the cable currents affected, the temperature change and the length of cable affected).

2.4 Additional Research

In addition to the instant and gradual failures described above, the following parameter changes could be investigated:

1. Different failure periods, for example: 6 months, 1 week or 6 hours.
2. Specific current ranges, for example: failure only detectable during peak cable currents (e.g. top 20%).
3. Different temperature changes, for example: +5%, + 20%, or + 10^{O}C.
4. Different special ranges, for example: 1 specific cable segment, or a 100m stretch.

Of these, the different periods and different temperature ranges are the most probable real-life failure scenarios.

Model	P-value
LSTM p1	0.01053
LSTM p3	3.052e-05
LSTM p1 c	0.0003615
LSTM p3 c	3.052e-05
LSTM AE1	3.052e-05
LSTM AE3	3.052e-05

Table A.9: P-vales from applying the Wilcoxon signed-rank test to the temporal models for instant faults. The alternative hypothesis used is that the AUCs of the linear regression model have a greater mean.

Model	P-value
LSTM p1	0.2388
LSTM p3	0.06984
LSTM p1 c	0.003357
LSTM p3 c	3.052e-05
Lin reg	0.009033
LSTM AE3	0.0003624

Table A.10: P-vales from applying the Wilcoxon signed-rank test to the temporal models for gradual faults. The alternative hypothesis used is that the AUCs of the LSTM-AE1 model have a greater mean.

Model	Encoding size	P-value
PCA	10	3.052e-05
PCA	25	3.052e-05
PCA	50	3.052e-05
PCA	100	3.052e-05
PCA	500	3.052e-05
PCA	1000	3.052e-05
PCA	2000	3.052e-05
PCA	4000	3.052e-05

Table A.11: P-vales from applying the Wilcoxon signed-rank test to the spatial models for instant faults. The alternative hypothesis used is that the AUCs of CAE4650 have a smaller mean.

Model	Encoding size	P-value
CAE	4650	3.052e-05
PCA	10	1 0.01113
PCA	25	0.1855
PCA	50	0.09072
PCA	100	0.5
PCA	500	1
PCA	2000	1
PCA	4000	0.5

Table A.12: P-values for the spatial models on the constructed instant faults. From Wilcoxon signed-rank test with alternative hypothesis that the models had lower AUC scores than the PCA model with 1000 components.

Model	Encoding size	P-value
PCA	10	3.052e-05
PCA	25	3.052e-05
PCA	50	3.052e-05
PCA	100	0.0003624
PCA	500	3.052e-05
PCA	1000	3.052e-05
PCA	2000	3.052e-05
PCA	4000	3.052e-05

Table A.13: P-values for the spatial models on the constructed gradual faults. From Wilcoxon signed-rank test with alternative hypothesis that the models had higher AUC scores than the CAE model

Model	Encoding size	P-value
CAE	4650	3.052e-05
PCA	10	0.0003624
PCA	25	6.104e-05
PCA	50	0.0002136
PCA	100	0.01508
PCA	500	0.03666
PCA	2000	0.1147
PCA	4000	0.02396

Table A.14: P-values for the spatial models on the constructed gradual faults. From Wilcoxon signed-rank test with alternative hypothesis that the models had lower AUC scores than the PCA model with 1000 components.

Model	P-value
PCA	0.1855
PCA c	0.1855
PCA LSTM	0.001258

Table A.15: P-values for the combined models on the constructed instant faults. From Wilcoxon signed-rank test with alternative hypothesis that the models had lower AUC scores than the PCA model with 1000 components.

Model	P-value
PCA	0.0008085
PCA c	0.0001526
PCA LSTM	0.0003624

Table A.16: P-values for the combined models on the constructed gradual faults. From Wilcoxon signed-rank test with alternative hypothesis that the models had lower AUC scores than the PCA model with 1000 components

Cecilie Haugsbø Hermansen

Using Deep Learning for Anomaly Detection in Distributed Temperature Sensing Systems for Submarine Export Cables

NTNU
Kunnskap for en bedre verden