NTNU

Norwegian University of
Science and Technology

Impact of Domain Knowledge in Time Series Forecasting of Oil and Gas Sensor Data

Øyvind Bratvedt

Sigurd Grøneng

Master of Science in Informatics
Submission date: June 2018
Supervisor: Jon Atle Gulla, IDI

Norwegian University of Science and Technology
Department of Computer Science

Summary

Production of oil and gas is a complicated operation, and due to this complexity, the work is being closely monitored. Sensors mounted on platforms measure the current state of the system, with alarms indicating when values are above or below a threshold that is considered normal. Values from the sensors are stored as time series, documenting the historical operation of the oil and gas platform.

The work in this thesis looked into how sensor time series can be utilized in terms of time series forecasting and anomaly detection. With the help of the deep learning technique long short-term memory (LSTM) neural network we have investigated how the domain knowledge of related sensors can affect the forecasts. Time series from sensors located on one of Aker BP's oil and gas platforms in the North Sea have been selected. In order to make sure that they are relevant, they originate from the same system on the platform. At a point in time, this system had a major malfunction, which makes it possible to see if this anomaly can be predicted by the LSTM. Subsets of the time series have been created based on their sensor type and location. These subsets have been used as an input to the LSTM in order to detect if some sensors have a high impact on the predictions.

Results show that, in terms of forecasting error, including a combination of related sensors can be better than both using all sensors in the input, and only the sensor that is predicted as an input. Some combinations can also greatly increase the forecasting error, suggesting that sensors should not uncritically be included. A large portion of the combinations ended up mimicking the time series, leading to low forecasting and anomaly detection power. Two combinations did not include the predicted sensor in the input, forcing the models to not mimic the predicted time series. These models had the highest forecasting error, but showed the most promising results of the combinations in terms of anomaly detection and forecasting power. Ultimately, we suggest extensive cleansing of data if deep learning should be used in this domain.

Sammendrag

Produksjon av olje og gass er en komplisert prosess, og på grunn av dette er all produksjon nøye overvåket. Sensorer plassert på platformen måler den nåværende tilstanden til systemet, med alarmer som indikerer når verdier er utenfor det som kategoriseres som normalt. Verdiene fra disse sensorene er lagret som tidsserier som representerer operasjonshistorikken til oljeplatformen.

Arbeidet i dette studiet har undersøkt hvordan tidsserier fra disse sensorerene kan bli brukt til å predikere verdiene fram i tid, og å oppdage avvik fra normal operasjon. Ved hjelp av *deep learning*-teknikken *long short-term memory*-nettverk (LSTM), har vi undersøkt hvordan domenekunnskap i form av relaterte sensorer kan påvirke prediksjonene. Tidsseriene kommer fra sensorer montert på en av Aker BPs oljeplattformer i Nordsjøen. For å forsikre oss om at sensorene er relatert til hverandre, kommer de fra samme system på plattformen. På et tidspunkt hadde dette systemet en større funksjonsfeil, noe som kan si noe om LSTM-modellen kan oppdage unormal systemfunksjon. Sensorene har blitt delt opp i kombinasjoner basert på hvor den er montert og hvilken type sensor det er. Disse kombinasjonene har blitt brukt som inngangsdata i LSTM-nettverket for å finne ut om det er noen sensorer som har større innflytelse på prediksjonene enn andre.

Resultatene tilsier at det å inkludere en kombinasjon av relaterte sensorer kan gi lavere prediksjonsfeil enn både det å bruke alle relaterte sensorer, og kun den sensoren som skal bli predikert. Noen sensorkombinasjoner kan også øke prediksjonsfeilen betraktelig, noe som tilsier at kombinasjonene må velges med omhu. Mange av kombinasjonene endte opp med å etterligne tidsserien, noe som gir en lav prediksjonsfeil, men en lite nyttig modell med tanke på predikering og deteksjon av unormale verdier. To av sensorkombinasjonene hadde ikke sensoren som blir predikert inkludert, noe som tvang modellen til å ikke etterligne tidsserien. Disse kombinasjonene hadde høyest prediksjonsfeil, men hadde de mest lovende resultatene av alle kombinasjonene med tanke på predikering og deteksjon av unormale verdier. Hvis *deep learning* skal bli brukt på dette domenet, foreslår vi omfattende rensing av dataen.

Preface

This thesis is submitted to conclude a Masters of Science degree in Informatics at the Norwegian University of Science and Technology (NTNU). The thesis has been supervised by Professor Jon Atle Gulla in collaboration with Cognite AS, and issued by the Department of Computer Science (IDI).

Acknowledgements

We would like to thank Professor Jon Atle Gulla for great supervision during the whole process of work on this thesis.

We would also like to thank the people at Cognite AS, more specifically Trygve Karper, Que Tran, Erlend Vollset, and Geir K. Engdahl for providing us with data, feedback, and support when needed.

Table of Contents

List of Tables

List of Figures

Acronyms

API	Application Programming Interface.
BarA	Bar Absolute.
ECML-PKDD	European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases.
ESV	Emergency Safety, Valve.
GRU	Gated Recurrent Unit.
IDI	Department of Computer Science.
ISA	International Society of Automation.
LSTM	Long Short-term Memory.
MAE	Mean Absolute Error.
MSE	Mean Squared Error.
NTNU	Norwegian University of Science and Technology.
P&ID	Piping and Instrumentation Diagram.
PCA	Principal Component Analysis.
PDF	Portable Document Format.
PI/PT	Pressure/Vacuum, Indicating Device/Pressure/Vacuum, Blind Transmitter.
ReLU	Rectified Linear Unit.
RMSE	Root Mean Squared Error.
RNN	Recurrent Neural Network.
SGD	Stochastic Gradient Descent.
SVM	Support Vector Machine.
TI/TT	Temperature, Indicating Device/Temperature, Blind Transmitter.
ZI	Position/Dimension, Indicating Device.

Chapter 1

Introduction

This chapter explains the background and motivation for the project, problem description, the research questions to be answered, and the chapter outline for the rest of the report.

1.1 Background and Motivation

Cognite[1] has received the task of digitalizing Aker BP[2], a Norwegian oil and gas company. This means to utilize the data generated by the platforms and create digital solutions that can improve the production and daily life on the platform. Oil and gas platforms are complex installations that require lots of equipment. In order to monitor the production on the platforms, sensors are mounted on equipment throughout the pipeline. Alarms are activated when a sensor value is not within a certain spectre that is defined as normal operation. What this approach fails to provide is predictive powers, since it will only alert when the operation has already transitioned into an abnormal state. By utilizing the historical data of the sensor, it is possible to investigate whether or not a predictive model of the sensor can be created. If it is possible to predict a certain amount of time into the future, anomalies and abnormal behaviour can be detected earlier. Detecting a potentially malfunctioning part, or a part that is about to fail as early as possible is important due to avoiding unexpected downtime and potential dangers failing equipment can cause. A

[1] https://cognitedata.com/
[2] https://www.akerbp.com/

financial benefit is also present due to the potential loss of production a process shutdown can cause.

1.2 Problem Description

The utilization of time series from sensors on the Aker BP oil and gas platforms has been limited. These time series could potentially be used to automate tasks for identifying when and which equipment that is malfunctioning, also for more complex states that earlier was assumed to be normal. While the data exists and is available, it is still hard to say which methods that are most appropriate and will give the most reliable results. There might also be major challenges and limitations within these that makes harder to solve. In this thesis, we will make use of these time series to investigate if the sensor values can be predicted, and if we can detect anomalous behaviour. We will use deep learning architectures, more specifically a long short-term memory (LSTM) neural network to do predictions. The main goal of the thesis is to investigate how inclusion of related sensors can improve the forecasts and anomaly detection in the time series.

1.3 Research Questions

The research questions to be answered in this thesis are listed below. They are mainly concerning the challenges of doing time series analysis of industrial data from the oil and gas sector.

RQ1: *What are the main challenges of time series analysis in the oil and gas industry?*
We will investigate how challenging it is to understand the time series, events affiliated with them, and how dirty the data is. We will also look into how applicable they are as a data source for forecasting.

RQ2: *How applicable is LSTM for time series anomaly detection in sensor data?*
How good a deep learning model like an LSTM network fits to this problem is also something we will investigate. Given the complexity of this model, how will it handle time series sensor data from the oil and gas industry?

RQ3: *How can domain knowledge of related sensors be used for anomaly detection in time series?*

By including time series from related sensors when forecasting, we want to investigate if this can improve the forecasts and anomaly the detection. This will also tell something about which sensors that impact forecasts the most.

1.3.1 Approach

We will investigate forecasting of sensor data from one of Aker BP's oil and gas platforms in the North sea. More specifically, the sensors are picked from a closed system on the platform. The goal is to evaluate how well deep learning models can forecast these sensor time series, both in terms of anomaly detection and forecasting on normal data. We will also investigate how the inclusion of domain knowledge can impact the results. That is why the sensor data is picked from the same system, since then it will be possible to test if including related sensors in the model will affect the forecast. This particular system of sensors also contains an anomaly, so we will also check if this anomaly can be predicted. We will specifically check the applicability of an LSTM network, and try to evaluate whether this approach can be used. Our focus is not on tuning and tweaking parameters in neural networks, but how different input data affect the performance of the networks. Acquiring a large enough clean industrial dataset can often be infeasable, something that also applied to our case.

When it comes to using deep learning on sequential data, material suggests using recurrent neural nets (RNN) because of their ability to capture dependencies in time [27]. RNNs can be used both as classifiers and for regression. Three widely used RNN approaches are vanilla RNN, gated recurrent units (GRU) [5] and LSTM [16]. There are studies that compare them, and they perform quite equally [34] [43], but LSTM is considered the most complex and can handle dependencies in time longer because of its architecture, which will be explained in Chapter 2. Therefore we chose to focus on using LSTM, and because we only have one labeled anomaly, LSTM as a regression model will be used.

1.3.2 Additional Work

In April 2018, we submitted a paper for the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases[3] (ECML-PKDD) regarding the impact of domain knowledge in time series forecasting of oil and gas sensor

[3] `http://www.ecmlpkdd2018.org/`

data. This paper can be found in Appendix A.

An additional experiment with the time series used in this thesis has also been conducted. We have used an autoencoder [39] to investigate how well another deep learning model performs on the same data. This experiment can be found in Appendix B.

1.4 Results and Conclusion

Results indicate that including time series from related sensors as an input to the LSTM can both improve and worsen the forecasting error compared to the baselines. Most of the time series combinations did to a certain degree forecast the most recently observed value, leading to low performance in terms of forecasting and anomaly detection. Two of the models had the predicted time series omitted in the input, forcing the model to learn more meaningful patterns. This led to high forecasting errors, but had the most promising results when looking at forecasting power and anomaly detection. Experimenting further with time series inputs omitting the predicted time series might further improve the applicability of the model.

1.5 Report Outline

The report is structured as follows: Chapter 2 presents the background theory for this thesis, and Chapter 3 covers the related work. The problem domain is explained in Chapter 4, while the data used for the experiments is presented in Chapter 5. Experiments and results are in Chapter 6 and 7, followed by discussion and conclusion in Chapter 8 and 9.

Chapter 2

Background

In this chapter, the theory behind the project is presented. Important techniques and terms from the fields of machine learning, time series, and neural networks are covered.

2.1 Supervised and Unsupervised Learning

Machine learning is divided in to multiple subcategories: The most used are supervised and unsupervised learning.

2.1.1 Supervised Learning

Supervised learning techniques create models based on pairs of input and output where the model tries to adapt to either classify the inputs into the correct output categories, or to predict the closest values to the outputs based on the inputs. Often the input is a vector and sometimes the output is too, depending on the problem. These models are trained with the help of an error measure which is usually a variant of the difference between the output of the model and the ground truth. The models are trained until the error between the model output and ground truth is below a set threshold [7].

2.1.2 Unsupervised Learning

Unsupervised learning is performed on unlabeled data. This technique is often used when there is no access to data that is labeled with the correct categories. This makes it challenging to evaluate the performance of the model because the evaluation is often done manually and on fewer samples than used when labeled data is accessible.

The way these techniques work is that instead of training the learner to predict the output Y, when getting X as input, it learns the distribution of the all the possible input data and uses that as a way to classify into a specific category or predict a certain value. Most techniques are based on statistical distribution or some kind of feature reduction [14]. Examples of unsupervised learning techniques are clustering [14], principal component analysis (PCA) [47], autoencoder [39], and deep belief networks [26].

2.1.3 Semi-supervised Learning

Semi-supervised learning is a combination of supervised and unsupervised learning. When the data consists of both labeled and unlabeled data, it is a semi-supervised problem. These problems are often present in the real world where for instance data is difficult and time consuming to label, thus only a small portion of the data is labeled [50].

2.2 Regression and Classification

There are mainly two modeling approaches where machine learning is used; classification and regression. Neural networks and other types of machine learning can be used for both classification and regression. Classification is defined as: ”Given a set of training data points along with associated training labels, determine the class label for an unlabeled test instance.” [2] In classification one separates between binary classification and multiclass classification. An example of binary classification can be classification of normal or abnormal operation of a system, based on values from the operation. Examples of multiclass classification is categorizing customers into different market segments, image recognition where there can be thousands of classes, or saying whether an operation is halted, normal or abnormal.

The output of a regression model is continuous values instead of a class. Typical use cases

are prediction of quantitative data like prices, temperatures or other amounts in general [10].

2.3 Time Series Terminology

Different terminologies for analysis of time series are presented in this section.

2.3.1 Random Walk

There are certain patterns in data occurring in multiple fields that are related to probability distributions. Brownian motion [15] and random walks, which loosely speaking can be seen as a one dimensional version Brownian motion, are examples of this. Since this movement is random it is thereby unpredictable. Still, it is only the movement that is unpredictable, and we often know that it is more unlikely to take a big step than a small step. If the step sizes follows a probability distribution it is often called a Gaussian random walk. Some types of data can be very similar to for example a random walk, even though every movement can be explained by certain real events. A typical example of this is financial stock data which is affected by supply and demand of the stock. In shorter periods like hourly samples for a time period of a couple of months, this can still be really similar to a random walk, which means that it is for all practical purposes is not predictable [32, 49]. What tends to happen when trying to model time series like this is that if the model performance is measured by taking the predictions distance from the actual value, it will always be best to guess the most recently observed value [19].

2.3.2 Autocorrelation

Autocorrelation describes the relationship of a variable to itself at certain distances, and in our context this means time. Positive autocorrelation means similar pairs, while negative autocorrelation means the opposite. Evaluating the autocorrelation of a time series can be used for identifying if its values have any correlation to values from earlier time steps. When there is no autocorrelation in a time series, it is just a collection of random values in some range. Random walks have a steadily decreasing autocorrelation. Clear seasonal trends can also show itself in autocorrelation plots [17].

2.3.3 Anomaly Detection

In anomaly detection, the goal is to detect data points or behaviour that deviates from what is considered normal. This could either be an outlier or an anomaly. The difference is that an outlier, even though it is distant from other observations, is considered normal behaviour or noise. Anomalies are those outliers that are interesting to analyze. Anomalies in time series can be defined as multiple values or patterns that deviates from the normal patterns, or values that are very different from other values [1].

2.4 Artificial Neural Networks

The majority of techniques presented in this section is based on *Deep Learning* by Ian Goodfellow, Yoshua Bengio, and Aaron Courville, 2016 [12].

The core component in an artificial neural network is neurons. The name is inspired by the architecture of the human brain that has neurons which in some ways can be considered similar to the artificial neurons. These neurons are essentially a functional mapping from a set of input to a certain output. By creating an input layer and an output layer consisting of a number of neurons and connecting these layers in different ways, they can create a very complex mapping. Input from one neuron to another neuron can be weighted differently, so some specific input can have more impact on the output than other inputs. The least complex neural networks is where every neuron in one layer in connected to every neuron in the next layer. If all the connections between neurons in the network always goes forward, it is considered a feedforward network. Another type of neural networks are networks with so called feedback connections, these are called recurrent neural networks.

Warren S. McCulloch and Walter Pitts introduced the research on neural networks as early as in 1943 [31]. However, it took around 40 years before the research caught substantial interest. Due to the computational complexity of neural networks, it was not until the recent years artificial neural networks became applicable to real world problems. Neural networks have been effective in applications like facial recognition [23], prediction of solar radiation [48], and detection of skin cancer [9].

2.4.1 Deep Learning

Deep learning is the use of artificial neural networks with at least two hidden layers, which are layers between the input and the output layers. This architecture became popular alongside with the increase of computational power during the previous two decades. The deep learning term was used to emphasize that the ability to train deeper neural networks now became possible, and research experiments with hidden layers were conducted [6]. Deep learning is often used in problems where finding relevant features and representing them in a good way is hard, for instance in processing of images, video, speech, and audio [24]. Deep neural networks are sometimes thought of as networks building its own abstractions from each layer. For example if the network is doing image classification one layer might identify shapes, and the next builds new abstractions based on the composition and relation of those shapes and so on. Figure 2.1 shows an example of a deep neural network with two hidden layers.

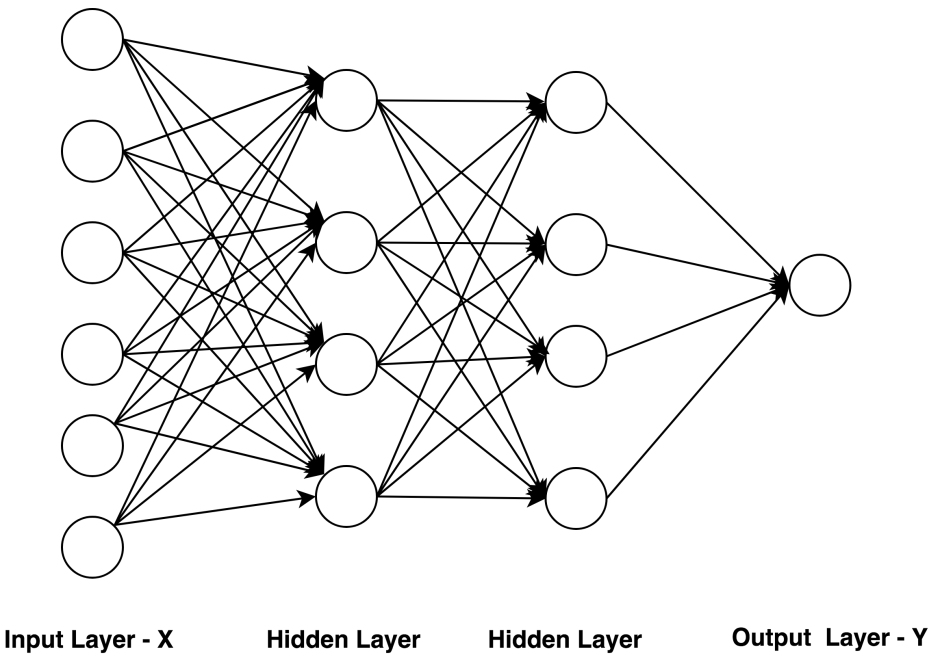


Figure 2.1: Deep neural network with 2 hidden layers

2.4.2 Recurrent Neural Networks

In recurrent neural networks the output of a layer is fed back into itself. This way time dependencies can affect the data without scaling up the network architecture. This makes it able to handle much longer sequences without increasing the number of weights significantly. This makes them especially good at handling dependencies between inputs and are therefore often used on sequential data, like understanding context in sentences [33]. Because the weights update more when training by going through time steps, they are even more vulnerable of vanishing gradient problem, like very deep neural networks. For example if one want to use the 100 last values of a multivariate time series with 10 dimensions, one can still use any number of neurons in the recurrent layer. The layer will iterate through the data 100 times, and each time it will use the output from the previous iteration in addition to the next one of the 100 values.

In Figure 2.2 we show two ways to think about a RNN. One is where an input vector is passed to the network cell and one part of the sequence is consumed each iteration for t iterations. If we unfold this it can be seen as t layers with the same weights where each part of the sequence is passed to different layers. The white box in the figure represents what we call the cell. These have a state which is based on the previous input. All layers except the first also passes their previous output to the next cell. Some variants of RNN divide the input to the next cell into more than one variable, but in the illustration it is shown as a single arrow. The vertical arrows represent the output. This is sometimes called the hidden state. Whether these are used depends on the wanted output of the layer. Any combination of input and output is possible. In the framework we are using one can choose if these values should be returned with a configuration parameter called ”return_sequences”[1].

Long Short-Term Memory Neural Network

Long short-term memory (LSTM) was first introduced by Hochreiter and Schmidhuber in 1997 [16] as an approach to avoid the vanishing gradient problem of RNNs. RNNs can remember patterns over short time periods, but as new information is fed to the network, the previously stored information vanishes. By introducing changes to the RNN architecture, the LSTM can remember dependencies in much longer sequences. Hochreiter and Schmidhuber claim it can remember dependencies in up to 1000 time steps. LSTM is constructed from memory cells with input and output gates, determining what to store in

[1] `https://keras.io/layers/recurrent/`

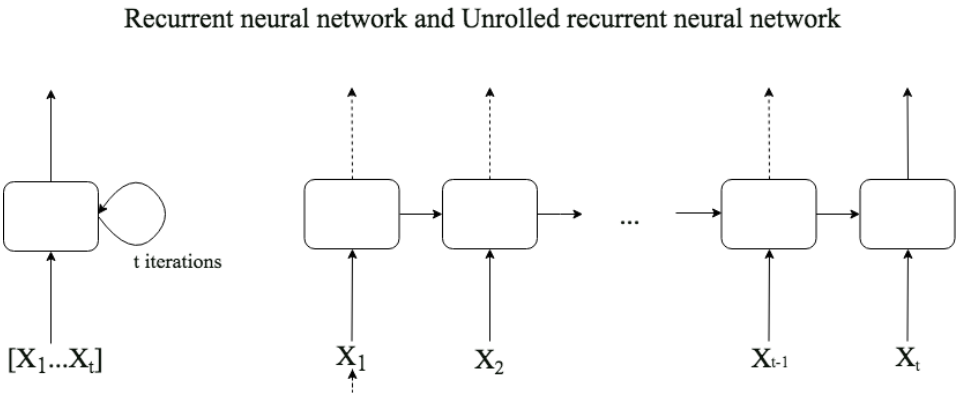


Figure 2.2: An unfolded RNN

the cell and what to forward to the next. Forget gates in order to reset the state of the cells was introduced by Gers, Schmidhuber, and Cummins, 1999 [11]. Without the forget gate, the values inside cells can grow without bounds and may cause the network to break. By removing the information that is not longer needed by the LSTM, the network performance increases. Evaluations of recurrent neural network in [18] suggests using LSTM when experimenting with recurrent neural networks.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.1}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.2}$$

$$\tilde{C} = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.3}$$

$$C = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.4}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.5}$$

$$h_t = o_t * tanh(C) \tag{2.6}$$

The equations above describe how the values of an LSTM cell are calculated [27]. Three of the equations are gates and use sigmoid activations which decide how much of a value is let through. These are the equations 2.1, 2.2, and 2.5. The result of these equations are multiplied with C_{t-1}, $\tilde{C}$, and $tanh(C)$ that are scaled to values between 0 and 1, where 0 is a fully closed gate, and 1 is a fully open gate. Equation 2.3 calculates the update to the cell value which is between -1 and 1. The new cell value C is calculated by adding

the value of the previous cell state multiplied with the value of the forget gate ($f_t * C_{t-1}$) with the current cell value multiplied with the input gate value ($i_t * \tilde{C}_t$). The output value of the cell is here noted h_t and is a tanh gated value of the current cell value.

Each of the LSTM neurons can act as a storing cell, because if the sigmoid of the values that controls the input gate are kept low, the value in the cell will not change very much until there is some input that opens up the gate for writing again. This differentiates the LSTM from other recurrent neural networks because there are no strong mechanisms to stop the stored value from being overwritten. The output gate controls whether the stored value is read from and used further out in the network, while the forget gate resets the value. An example of a cell is shown in Figure 2.3. The values that goes into the cell horizontally controls how the input and output values flow.

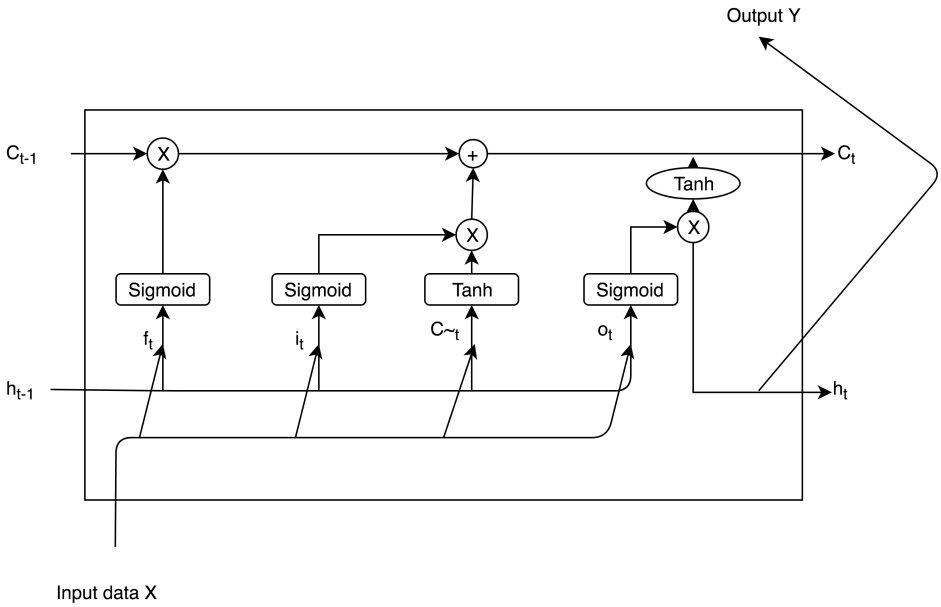


Figure 2.3: LSTM cell

2.4.3 Gradient Descent

Most training algorithms for neural networks are based on gradient descent. Gradient descent works by finding out how the weights in the neural network should be adjusted to get a result that is closer to the ground truth. The difference between the predicted values

and the actual values can be computed using a chosen cost function like the mean squared difference between actual and desired values over all neurons in the layer. Each weight can be either positively or negatively adjusted and the amount it is adjusted with can be small or big. How much each weight should be adjusted to get lower cost is computed by finding the gradient of the cost function for each neuron in the current layer. The output of a neuron can not be adjusted directly, but can be influenced by adjusting the weights into that neuron. If there is more than one output in the output layer this is done for each neuron and the adjustments are added up, which means some of them might cancel each other out. This whole process is repeated for each training example. This canceling out can also happen when updating the weights backwards in the layers. The initial values of the weights are often set randomly, but some networks requires special strategies when initializing to not get stuck, or to get better results faster [12].

2.4.4 Optimization

Because computing the exact direction of a gradient can be very computationally expensive, one often only takes a random subset of the training examples to compute the general direction before applying it. This can give an approximation that is good enough while using much fewer calculations. These kinds of methods are commonly known as stochastic methods. Training with smaller batches made from samples of the whole training set can also give a generalizing effect which prevents overfitting.

Stochastic Gradient Descent

Stochastic gradient descent (SGD) is a straightforward gradient descent approach which uses random samples for finding a gradient to update its weights. The updates are not only controlled by the gradients but also multiplied with another parameter called learning rate. This is a parameter that decides how far the neural network optimizer can go towards a minimum each training batch. If set too high, areas with lower cost might be stepped over, and if set too low it might get stuck in areas with low curvature. For example if some area is flat, which means that no direction will give lower loss, then the gradient is zero. SGD have been used to handle machine learning on a large scale where an optimization algorithm of low time complexity is required [4].

Adam

The Adam [20] algorithm is one of the algorithms that spring out of the idea of automatic adaptive learning rates. It is similar to other algorithms that tries to automatically adapt learning rate, but also adds momentum to the parameters before calculating the gradient. Like AdaGrad [8] and RMSProp [38] it decreases learning rate proportionally to how much the gradient changes direction between iterations. The momentum is not only built from the length of the gradient but by accumulating each subsequent iteration in approximately the same the direction. If the gradient suddenly changes direction a lot while the momentum is high, the algorithm will keep going in about the same direction but start to tread more lightly. It requires several updates in a new direction to change the direction of the momentum. The most important distinction between AdaGrad, RMSProp and Adam is how the momentum is accumulated. Adam is said to perform well, but it has been hard to explain theoretically why it does perform good.

2.4.5 Loss Functions

A loss function, also called cost function, is used to measure how far from the ideal values an output is. During training of neural networks, these functions are used to give the model a feedback on how far the away predictions are from the ground truth. Since we are not using classifiers in our experiments we will only go through the regression loss functions. All notations assume that error e is $e = x_t - x_p$ where x_t is the true value and x_p is the predicted value. In these functions t means the error at time t, while n is the number of samples to evaluate.

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|e_t| \tag{2.7}$$

The mean absolute error (MAE) (Eq. 2.7) function weights all errors linearly which means that some large errors can be allowed if the rest are small.

$$MSE = \frac{1}{n}\sum_{t=1}^{n}e_t^2 \tag{2.8}$$

What separates the mean squared error (MSE) (Eq. 2.8) from the MAE is that it pun-

ishes large errors much harder since they grow exponentially as they stride away from the training values.

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2} \tag{2.9}$$

The root mean squared error (RMSE) (Eq. 2.9) is the square root of the MSE. By taking the root, it is easier to interpret the error since it does not grow exponentially.

2.4.6 Activation Functions

Activation functions are used in each neuron of a neural network. They are applied after all the input to a neuron is summed and gives out a new values in a certain range. They decide how active a neuron is. Some of them are called squashing functions because they squash a value in any range into a new range often either between 0 and 1, or -1 and 1. Activation functions are often not linear when dealing with multilayered neural networks [41]. Next we will explain the three most used nonlinear activation functions.

Sigmoid

The sigmoid is the most used activation function historically. It can be seen as a representation of the firing of a neuron where 0 is not firing at all and 1 is full activation. The biggest drawback of this function is that it suffers from the vanishing gradient problem. When the combined input to the activation function becomes very high (close to 1), or very low (close to 0), the gradient will become very small and these values will sometimes be multiplied into other calculations in different ways depending on the optimization algorithm that is used. Figure 2.4 shows a plot of a sigmoid function from -5 to 5.

Tanh

The tanh function is equal to a scaled version of the sigmoid function. It is often used instead because it is zero-centered. This can be a problem when all inputs to a neuron are positive. It is said that when the output from neurons can be both negative and positive

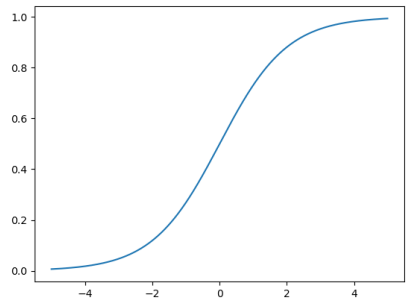


Figure 2.4: Sigmoid

it gives a more efficient update to the weights [25]. Still, this function also suffers the vanishing gradient problem [42]. Figure 2.5 shows a plot a a tanh function from -5 to 5.

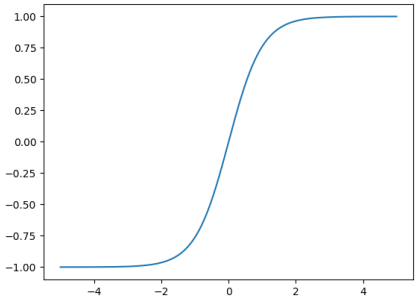


Figure 2.5: Tanh

Rectified Linear Unit - ReLU

Both the sigmoid and tanh functions can suffer from vanishing gradient. Because the ReLU does not converge to certain values when its input gets higher, neither will the derivatives. On certain problems, using ReLU seems to be able to make the *error* converge many times faster than with sigmoid and tanh alternatives [22]. Still, the ReLU has another problem: When the input first gets below zero, the gradient will always be zero, which it hard to recover from, and the cell can ”die”. There are versions of ReLU which tries to handle this problem by having a small slope instead of a constant zero when the input is below zero. This is called leaky ReLUs.

ReLU or any activation function that do not saturate should not be used for the internal gates in LSTM cells because they might ”explode”. It does not make sense that a gate is more than 100% open. Figure 2.6 shows a plot of a ReLU from -5 to 5.

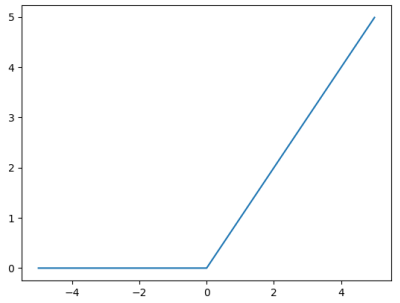


Figure 2.6: ReLU

2.4.7 Dropout

Overfitting is a common problem in neural networks, affecting the performance negatively. This can be because the dataset in too small to learn that some features should be ignored, or the network learning all the features too specific. If there are anomalies or wrongly labeled samples in the dataset this could also be an issue. Depending on how hard the errors are punished, the model might find a very complex solution to adapt to these outliers. A common solution is regularizing the network using so called dropout. Dropout deactivates a number of random neurons for a certain training batch. One of the most common deep learning frameworks, Keras [2], allows each layer to have dropout, and takes a fraction as the parameter. This parameter is the fraction of neurons in that layer that is deactivated. Neurons are only deactivated during training, and not during prediction or generation [42].

[2] `https://keras.io/`

2.5 Evaluation Methods

2.5.1 Evaluating Unsupervised Methods

It can be very hard to evaluate unsupervised methods since like mentioned, these techniques are often used when there are no labeled data available. Often one need to label a unseen subset of the data to measure against. This assumes that the subset is representative for the whole set. Generative models like autoencoders can be easier to evaluate. Because they try to recreate the input, one can just measure the error between the input and the output. There are still many cases where a very low error means that the model is overfitting which can make it much less useful.

2.5.2 Evaluation Metrics

The loss functions mentioned in Section 2.4.5 can also be used as evaluation functions for regression problems [30, 37]. They provide a representation of the error as a single number. The most used is MSE, punishing predictions that are far away from the actual value. If equal punishing of the errors is desired, the MAE is an alternative to the MSE [45]. For comparing results over different datasets where the values have a different range, a percentage version of these can be used.

Chapter 3

Related Work

In a summary article of RNN from 2015, *A Critical Review of Recurrent Neural Networks for Sequence Modeling* [27] they describe the work where these types of networks do exceptionally well. Like mentioned in Chapter 2 this is in processing audio, speech, video, and text, but this article does not really mention any work on forecasting or regression like we need to do on our data.

There is a limited amount of articles that do forecasts on industrial sensor data from oil and gas platforms with LSTM. The reasons for this might be that they are not published for the public to see, or that there is limited research on the topic. There still are numerous previous studies that use LSTM networks to do forecasting of sequenced data from other domains. Most of the articles relevant to our work focus on establishing good prediction models, before using them to do anomaly detection. In Chapter 8 we will compare this to our work, and include metrics from the related work.

3.1 Long Short Term Memory Networks for Anomaly Detection in Time Series

In the paper from [35] they do forecasts on one time step using an LSTM, measure the forecasting error in an n-dimensional space, and transforms it into a Gaussian distribution

in order to detect anomalies. The most suitable threshold that gives the best precision and recall is then calculated. Their experiments use four different datasets where three are publicly available[1], and one industrial dataset that is closed for the public. The public datasets consists of measurements of electrocardiography readings, power consumption and sensors from space shuttles, while the private dataset is engine sensor data. Since the patterns in the data are periodical, they can be detected relatively easy. The dataset that is the most challenging to do anomaly detection on is the engine data, according to the authors. On the space shuttle and the power consumption time series, the LSTM clearly performs better than classic RNN, and on the engine sensor time series they are about equal with the RNN doing slightly better, with 0.4 improvement in precision but 0.2 reduction in recall. They use three different LSTM architectures one for each dataset. One of them with a single hidden layer and two with two hidden layers, with the following number of neurons: 25, (30,20), and (35,35). They all have about the same performance on precision, but on recall the (30,20) architecture does significantly better and the single layer does somewhere in between. We think that this is more because of the kind of data than the architecture after inspecting the plots visually. The recalls of their models are low, 0.2, indicating that that the best performing model is only catching up to 20% of all the anomalies. This counts for both RNN and LSTM with recall of 0,19 and 0,17 respectively. The precision is 0.93, 0.94 and 0.94 in the same order.

3.2 Temporal Modelling of Bug Numbers of Open Source Software Applications Using LSTM

Prediction of bugs with the use of historical data has been experimented with in [36]. This paper predicts bugs from three different sources, Debian, Eclipse, and Mozilla, represented as time series indicating number of bugs at each time step. Their LSTM network is made by one hidden layer with four neurons. Three methods of creating input into their models are tried on the three datasets, and all of them predicts one value. The first of them is predictions based on windows of earlier values, the next is by doing this with values from the other bug time series. The last one also adds the values of the covariance between the other values that builds the input vector. They say the input consists of 2 lags, and we have interpreted this as a sliding window of size 2. By including data from another time series into their models they achieve somewhat better results. Slightly better results

[1] `http://www.cs.ucr.edu/~eamonn/discords/`

were also achieved when including covariance of the included time series in their input. However, their results seem to be affected by time series mimicking which is covered in later chapters. Their results for the three different datasets vary very much because the value distribution of their datasets are so different. The three ranges of 20-100, 50-350 and 500-2500 gives at best RMSE of 14.92, 67.44 and 210.86 respectively.

3.3 Applying Recurrent Neural Networks for Multivariate Time Series Forecasting of Volatile Financial Data

This paper [43] tries to predict the price of Bitcoin with the use of LSTM and GRU networks among others, and compares the performance of the models against each other. Their dataset only consists of 1615 records, but the number of features in each record is 293, which is high. These features include the values of different cryptocurrencies, Google Trends, and miscellaneous financial data. With this high amount of features, one can be quite certain that not all of them are of relevance. This is an interesting approach, since it includes so many features. We do not think all of them are relevant at all times, and question if this will confuse the models. All their results are achieved with very little training time, which usually means running training for fewer epochs or having limited data to train on. The GRU and LSTM have a training time of 83 and 43 seconds respectively. These measurements are when training on what seems to be a laptop machine with one CPU with four cores and an dedicated NVDIA QUADRO M1000 (middle end performance video card[2]). This will most likely affect the precision of their forecasts. The price of Bitcoin when they performed predictions was between 3000 and 5000 USD which will give higher absolute error than in other related work. The RMSE is 74507.16 ,75686.96, and 65484.81 for LSTM, GRU, and ARIMA-DR respectively. When inspecting the plots it seems to be a mix of mimicking the curve (guessing previously observed values) and learning correlated features.

[2] `https://www.videocardbenchmark.net/gpu.php?gpu=Quadro+1000M`

3.4 Prediction of Temperature in Hot-axles of Locomotives

In the article from [29] they used LSTM to predict temperatures in train axles because this is directly linked to operational status of trains. They say that by predicting these values one can check the deviation from the actual values and use that as an indication of operational status instead of using the temperature directly. To perform predictions they use an LSTM network with one hidden layer of 64 neurons. As input to the LSTM network they use 13 sensors located on trains, where 8 measure temperatures in the axles and related areas, 3 of them measure pressure, 1 measures train speed, and 1 pipe flow. These values are sampled at 10 second intervals. They use sliding windows of size 16 based on their experiments not included in their paper. With these values they predict 1, 6 and 12 time steps in 6 sensors placed in the train axle. Since they only provide max error from their predictions, it is not obvious how to compare their results to ours. As one may expect, their max predictions error gets bigger the more time steps they predict. Still, the error is not particularly large. For 2 minutes forward in time they have a max error of 1.9 degrees, which is not even the double of their best 10 second prediction. We still do not know what these values normally are and exactly how their testing set looks. Since they use linear interpolation on their data to sync the data at 10 second intervals, it could mean that all spikes are smoothed out if the original sampling is size more frequent than 10 seconds.

3.5 Comparison of Sample Size

The importance of data versus algorithms has been a discussion among the data scientists for a time. The intuitive explanation of why the amount of data is so important, is that the more data there is, the more representative it will be for the general problem. An article from 2001 shows that all results seems to improve with bigger datasets independent of the algorithm [3]. In another article from researchers at Google [13], they write about how bigger datasets in the text and speech processing domain enabled the use of models based on statistical correlation instead of annotated features. They write that annotations are not needed because these can be inferred statistically if there is enough data. Because they have so much data, the inferred correlations between words can be even more efficient than manually labeled data. Keep in mind though, that the domain they write about is very

different from ours.

Table 3.1 is a listing of all the time series in related work where the number of samples is defined. As a comparison, the number of samples in the time series used in this thesis is also included. There are big differences in the size of the datasets in our experiments,

Table 3.1: Size and number of time series in related studies

Data Set	Samples	Number of Time Series
ECG - 1 [35]	3750	3
ECG - 2 [35]	5400	2
Power demand [35]	35040	1
Space shuttle [35]	5000	1
Engine dataset [35]	Unknown	Unknown
Debian [36]	155	1
Eclipse [36]	155	1
Mozilla [36]	155	1
Bitcoin predicting set [43]	1615	293
Datasets in this thesis	190000 (10 min)	1-24 (depending on experiment)
Datasets in this thesis	35000 (1 hour)	1-24 (depending on experiment)

3.6 Summary

There have been some experiments using LSTM for regression problems, but we still think there is room for more exploration. In the anomaly detection article [35] we think there is missing some details of why the recall was so low and how the actual predictions looked. The axle temperature prediction [29] experiment is very similar to ours because they mostly uses the same kind of data, temperature and pressure, but unfortunately they do not provide enough metrics and plots to evaluate the models. Most of the experiments also predict only one time step forward, which means that the models can easily go into the pitfall of just predicting the most recently observed value instead of learning meaningful features. We suspect this may be the case for Bitcoin predictions [43] more than in the other predictions because these data are so unpredictable. Surprisingly, none of these articles mention that this might happen. In our experiments we will compare both one step predictions and six step predictions to try to find out what is really going on. Even though the Bitcoin prediction experiment utilizes a wide range of inputs to the model, it is hard to

say how correlated they are and whether this will confuse the model.

Chapter 4

Problem Domain

This chapter will explain the relation between Cognite and Aker BP, the data generated from the platforms, and methods to utilize this data.

4.1 Cognite and Aker BP

Cognite and Aker BP have a close relationship in terms that their mission is to digitalize Aker BP's day-to-day business. Operations on an oil platform generates lots of data of different types and from different sources. During the years of platform operation, the utilization of this data has been limited. The mission for Cognite is first and foremost to make use of this data with the help of machine learning, artificial intelligence and augmented reality. Another comprehensive task is to structure all the data into one data platform. Since they come from different systems, all with different formats, merging the data together is cumbersome work. Also, the amount of data to store and analyze is on a very large scale, so choosing the correct software and hardware infrastructure is important.

4.2 Data Sources

This section presents the different types of data and the data sources that are fetched from the platforms.

4.2.1 Work Orders

Work orders describe work that has been done on some equipment on a part of an oil platform. This includes scheduled maintenance and work done because of malfunctioning equipment, whereas the latter might be the most interesting for our experiments. Each work order contains start time and end time in epoch timestamp. Often these values are no more specific than a date, and start and end might time be at the same date. Some of the work orders are missing start or/and end values, and are set to 1. January 1900. Other fields of interest are the description, text, and priority of the work order. Table 4.1 shows an example of a work order and the most important fields. Names, companies, and IDs have been anonymized.

Table 4.1: Work order with important fields included

Fields	Value
startTime	1240437600000
endTime	1271887200000
description	Sea water leak in port thruster room
MaintenanceType	Corrective
Priority	3
Text	Sea water leak in port thruster room 23.04.2009 09:24:45 John Doe Inlet pipe on cooler is leaking in velding. Needs to be replaced, piece of rubber placed on leak to prevent water running out. Inlet and outlet valves are not closing properly, and also needs too be replaced. This job has be handed over to company x. Job postponed. Richard Doe Changed date as work center had not been changed , and safty level is 1. 29.03.10 John Doe had a meeting regarding leaking pipe and company y working with this issue. Deadline postponed. Richard Doe
assetIds	[]
FunctionalLocation	(ID of equipment)

The *startTime* and *endTime* defines the start and end of the work order. In this case, there is almost a year between start time and end time, which makes it challenging to know when exactly the work was performed. *MaintenanceType* and *Priority* are important fields since they indicate whether or not the work order is corrective or preventive, and the priority of the work order. In this case, the maintenance type is *Corrective* which indicates that something has malfunctioned and needs to be fixed. This is the kind of work orders that are interesting. The priority of the work order is 3 on a scale from 1 to 4, which indicates that it is not of high priority. The *Text* field shows that multiple people have been working on the order, thus the text being a bit unstructured. Finally, the *assetIds* and *FunctionalLocation* fields are important since they are defining the IDs of the equipment that is under maintenance. However, these fields are often not defined, making it challenging to find out which equipment that is affected.

4.2.2 Sensor Time Series

Sensors on the platforms log its values, creating time series of the sensor history. Each row in these data contains an epoch timestamp and one sensor value. It is these values we are trying to predict and use to detect abnormal behavior. The time interval between the samples are not always consistent, both between different time series and within one. This means that if several sensors should be used as features, their timestamps must first be synchronized. Also, when looking at single sensor values, data must either be interpolated, or the distance in time since last measured value must be part of the features of the model. Sometimes it seems like the equipment might have been turned off because values are missing for longer periods. Some sensors do not sample on a stable rate. Table 4.2 shows how a time series from a sensor is represented.

Table 4.2: Example of a time series

Timestamp	**Value**
1396603740000	21.4415263837243
1396604940000	21.17951122794965
1396605540000	21.454576614313016
...	...
1517389740000	29.431882159967618
1517390340000	28.71654433309158
1517390940000	27.821290478590058

4.2.3 Time Series Metadata

Each time series has information affiliated with it, like typical values, data range, and some of them contains descriptions of what the sensor is measuring. An example of the a time series metadata entry with its most important rows is in Table 4.3. The name of the sensor has been anonymized.

Table 4.3: Selected rows from time series metadata

Metadata	**Value**
Name	(name of sensor)
Descriptor	Production (PPT) Pressure B08
Unit	BarA
Span	100
Typical value	50

The fields are very inconsistent and many has missing or wrong values. As of the case in Table 4.3, the *Span* and *Typical value* field are not correct since the values span over more than 100 values, and the typical value in this time series is a lot higher than 50. This is the case for many entries, where either the data is missing or wrong, making it challenging to utilize.

4.2.4 Piping and Instrumentation Diagram

Piping and instrumentation diagram (P&ID) documents the functionality of the platform. It is an overview of the pipeline and all the equipment that is used in the process of producing oil and gas. The symbol notation of a P&ID is mainly based on the ISA 5.1 standard[1]. In this context, P&IDs can be used to find relations between equipment and sensors by manually examining them. The format on the P&IDs are either native PDF or images that has been scanned and saved as PDF.

[1] `https://www.isa.org/isa5-1/`

4.3 Utilization of the Data

Many of the time series contain a sudden increase or decrease of values. The reasons for this can be many: It can simply just be a faulty sample reading from the sensor, failing equipment, turned off equipment, or just normal operation. This makes it challenging to filter out noise, and distinguishing normal operation from abnormal operation. We thought of some techniques with incorporation of other data sources in order to deal with this:

4.3.1 Using Time Series Metadata

When looking through a number of time series metadata we found that most of them have *Typical value* 50 or 0. By inspecting the actual time series we found that this is generally not the case. If this had been correct, using the *Typical value* field in time series metadata and checking against the values in the time series, the values that are very distant from the typical value can be labeled as an outlier. The problem is that it is not possible to know what is causing the outlier, and since typical value in time series metadata often is not correct, this is not a very reliable technique.

4.3.2 Matching Work Orders with Time Series

All maintenance that is performed on the platforms is logged in work orders, so this is a natural place to start looking for events that cause anomalies in the time series. If enough anomalies are found, this can be transformed to a classification problem rather than a regression problem. The first step is to match time series with the work orders. Since they do not have a direct mapping to each other, the time series needs to be matched with the ID of the equipment the sensor of the time series is mounted on. Then the time series can be matched with the work order if the work order has the *assetIds* field specified. There is a limited amount of time series that can be mapped to work orders, because the lack of specification of the *assedIds* field in the work orders. Some work orders also specify the equipment ID of the sensor in the *Text* field, so they can be matched that way as well. The next challenge is the quality of the data in these related work orders. If the *MaintenanceType* is *Corrective* the work order is of interest, since that indicates that something has gone wrong and needs to be fixed. However, this does not necessarily mean that it can be seen in the related time series. Not all equipment have sensors mounted

on it, and most of the bigger equipment have multiple sensors affiliated. If the affected equipment in the work order is for example a valve, this could propagate to other sensors as well. Which sensors that are affected is not specified in the work orders. There would have to be a mapping from each sensor to every other related sensor in the data to infer where the errors may propagate. Even then they would probably have to be manually checked before they can be used as ground truth for the training algorithms.

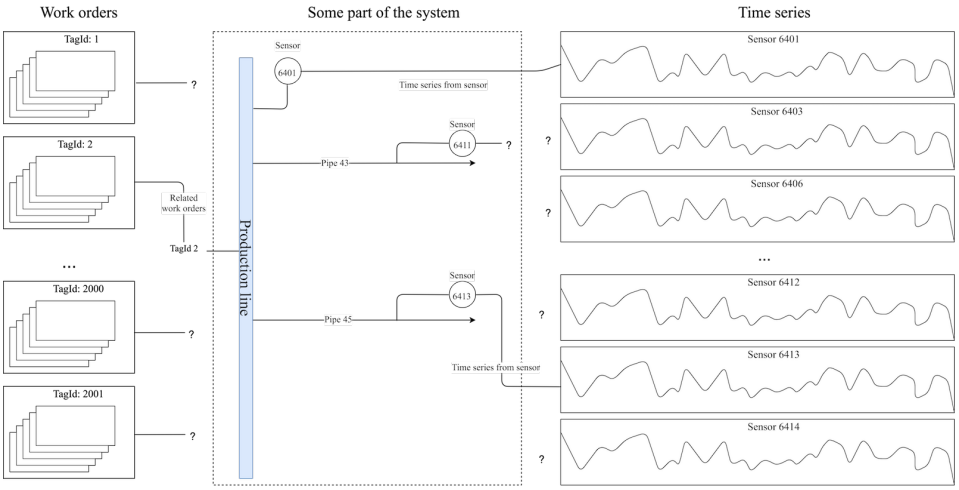


Figure 4.1: Visualization of the work order time series mapping problem

In Figure 4.1 we illustrate our situation. We have the work orders, P&IDs, and the time series, but we do not know how the relation between them without manually going through every P&ID. We only managed to map the time series to a work order if they were linked to the same equipment. For example like in the figure, we might find out that time series 6413 is from a sensor mounted on pipe 45, but we can not know that this mounted on the production line which might have many work orders.

As mentioned in Section 4.2.1, the start and end time of the order is often not specific enough or is missing, making it difficult to determine exactly when the work order was performed, not just when it was logged. We have performed some case studies where we have marked the start and end time of work orders that we have managed to relate to a time series. Figure 4.2 illustrates one of these time series in the case study. The green vertical line is both the start and end time specified in the work order, and by visually inspecting the time series at that timestamp, no apparent anomalies or outliers are present.

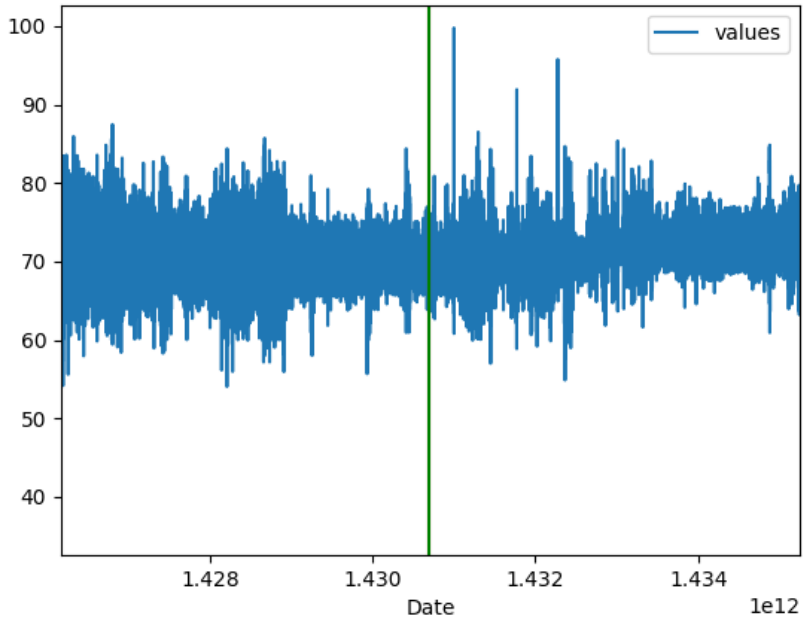


Figure 4.2: Time series with marked start of work order

Relating all work orders with time series like this is cumbersome work without any particular profit. Most of the work orders are not detailed enough to really understand the context without help from a domain expert.

4.3.3 Relate Time Series in Close Proximity

A potential approach to get better forecasts is to include related data in the input of the forecasting model. By including time series from sensors in the same system and in close proximity of the sensor to be predicted, the models can learn patterns that are not possible to learn when only using the single time series as an input. These related sensors can be found using the P&IDs, but domain knowledge is needed in order to verify whether or not they are closely related. This can be confirmed with the help of a domain expert, which is something that we have utilized in the experiments in this thesis. Domain experts can also help selecting systems with known anomalies, to check if the predictions models are able to detect them. By including the related time series, we can investigate if they can help

identifying the anomalies at an earlier stage.

Chapter 5

Data

This chapter explains the data that has been used in our experiments.

From this chapter and out we will be writing about a specific anomalous incident that happened during the period our dataset is from. This incident, hereby referred to as ”the anomaly”, is separate from the general concepts of anomalies and anomaly detection. When we write ”the anomaly”, it describes an a incident that is much more prevalent in our data than other possible anomalies in our dataset. We are sure our dataset contains many anomalies, but we have not matched the documented reports of these to our dataset because of reasons described in Section 4.3.2. Some of the seemingly anomalous behaviour seen in the sensor plots can be explained by testing of equipment and temporary changes in production in order to change or fix equipment.

5.1 Data Selection

With the help of a domain expert, a total 24 sensors originating from one particular piping system on one platform were selected for our study. The main reason for selecting these time series is because they are related, and that the system went into a malfunctioning state in the middle of 2017. This provides us with a known anomaly, opening the opportunity of evaluate how good our models are at anomaly detection.

The sensors selected have been sampling values regularly from 5th of March 2013, providing us with data spanning over approximately five years. There are four different types of sensors in the system: PI/PT (pressure), TI/TT (temperature), ESV(emergency safety valve), and ZI (choke indicator). PI/PT and TI/TT sensors are quite straightforward to understand since they are measuring the pressure and temperature of the pipeline. ESV sensors have the value of 0 or 1, indicating if the emergency safety valve is closed or open. These valves are automatically releasing pressure when the temperature or pressure is too high. Finally, the ZI sensor is mounted on a manually adjustable valve that controls the amount of flow through the whole pipeline. Adjustments to this valve will propagate backwards in the system and affect the values of sensors located prior to this valve in the pipeline. Since this valve is adjusted by humans, predicting it is not possible. Table 5.1 shows how many sensors there are of each type and the unit they are measuring.

Table 5.1: Sensors fetched from the platform

Sensor Type	Unit of Measurement	Number of Sensors
Pressure	BarA	7
Temperature	Degrees Celsius	7
ESV	Closed valve: 0, open valve: 1	9
Choke indicator	Percentage of how open the valve is	1

We chose one PI/PT sensor and one TI/TT sensor to forecast, located approximately in the middle of the system. These sensors were chosen since it was in these time series where the anomaly was most apparent. Figure 5.1 illustrates an overview of where the sensors are located in relation to each other.

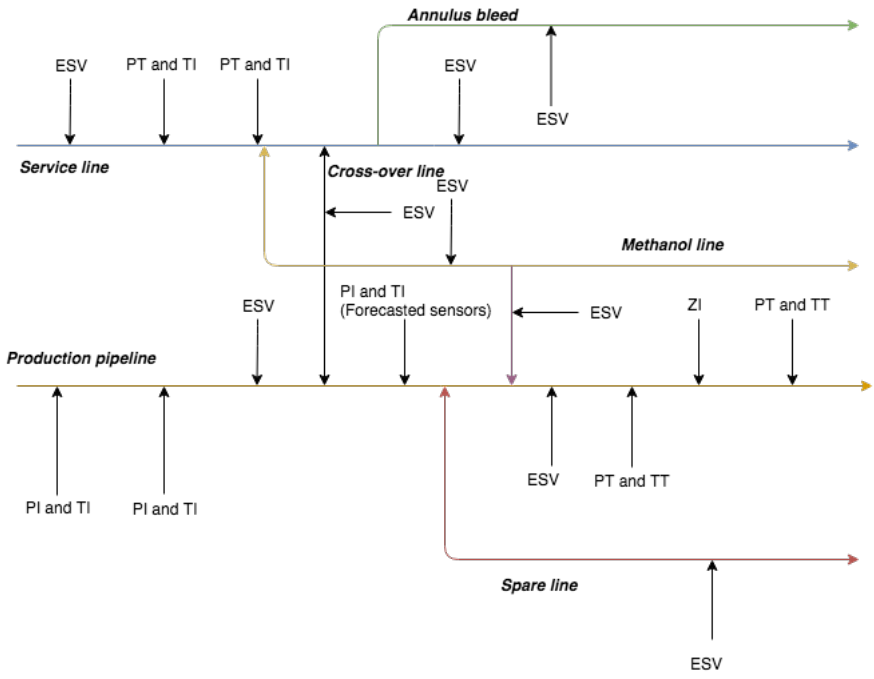


Figure 5.1: Sensor network

Figures 5.2 and 5.3 show plots of the time series that we are trying to forecast in the experiments. The first section, marked in red, contains the anomaly of the system, which can be seen with the huge drop of values in both time series. The second section, marked in green, is when production goes back to normal. These two sections of data will be used as test sets in our experiments, and will be further explained in Chapter 6.

As can be seen in both plots there are fluctuations where the values go very high or very low. Which of these that can be explained by equipment tests, system pauses for changing installations, or if they are induced by actual malfunction is unknown. In other words some of these events are controlled by user input to the system which makes them hard, if not impossible to predict. Because there are so many of these we do not doubt it will affect the training of the models, but we hope that the amount of data will compensate for this. An increase of pressure can be seen in Figure 5.2. The reason for this is that as the oil reservoir gets drained, more pressure is needed to extract the oil from the reservoir.

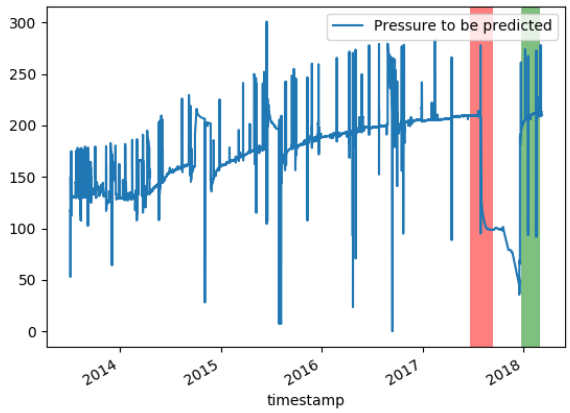


Figure 5.2: Plot of the pressure sensor we are trying to forecast

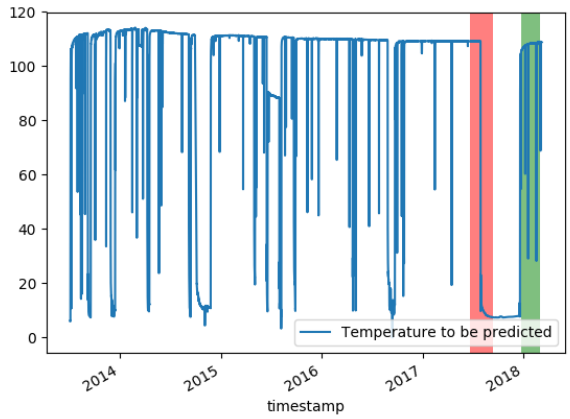


Figure 5.3: Plot of the temperature sensor we are trying to forecast

5.1.1 Faulty ESV Values

All data were fetched via an API offered by Cognite. It turned out that the ESV time series we received from the API were wrong because of how they were aggregated and interpolated in the API. To represent the ESV's actual state this should be done in specific

way called step interpolation. We knew the sample rate of some of the sensors because it is included in the name of the time series. Most of them were sampling every minute, something we assumed also was done for the ESV sensors since they also contained suffixes like ”10sSAMP”. When plotting them we saw that there always were linear lines over longer periods which we thought was very unnatural. We later found out that they only registered when they changed and not at fixed time spans. Because the API used averaging and interpolation to create all values needed to fill in between two points, most of our ESV values are incorrect. This means that while the values we received were linearly decreasing or increasing, the correct values were actually straight persistent lines until the value drops or raises.

5.2 Autocorrelation

A method to detect patterns or checking the correlation of the data is by inspecting the autocorrelation plot of the time series. This shows how correlated values are over all possible lags. Figures 5.4 and 5.5 shows the autocorrelation plots of Figures 5.2 and 5.3. A slowly decreasing linear curve means there is a strong autocorrelation. For at least the first 10,000 time lags this seems to be the case for both time series.

The reason we inspect this is to see if there are any sign that this resembles a random walk. If this is the case, the only thing we can surely say in the short term in that there is a low probability that the values will move very much from one step to the next. The strong correlation in the start can therefore indicate that the data is very hard to predict, because the only pattern found is that the values does not move much from the previous time step. Although the correlation line rarely goes between the dotted no-correlation lines (located very close to 0 on the y-axis), this analysis is otherwise not very promising.

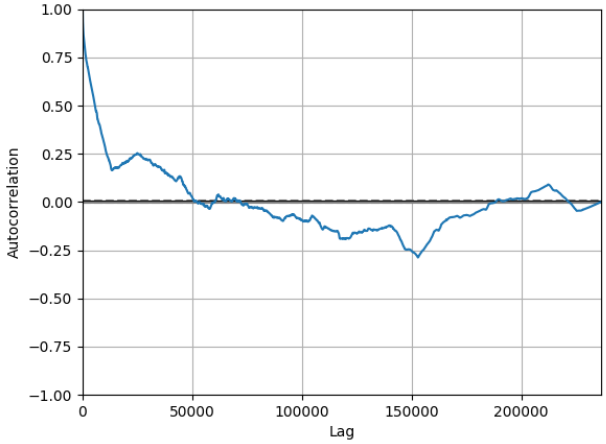


Figure 5.4: Pressure autocorrelation

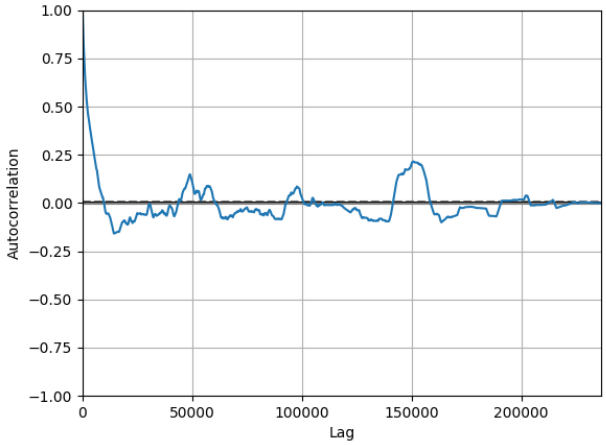


Figure 5.5: Temperature autocorrelation

Chapter 6

Experiments

This chapter explains the experiments conducted and the methods used in the experiments.

The main goal with the LSTM experiments was to investigate how using data from different sensor combinations can affect the forecasting results. By creating 12 different combinations based on knowledge of the pipeline system, results of these combinations can be compared against baselines. The two baselines are the forecasts with all the sensors as an input and only the single sensor that is forecasted as an input. The combinations were all trained to forecast two sensors: a pressure sensor and a temperature sensor. These combinations were trained on data with 10 minute interpolation and 1 hour interpolation, with predictions of one time step and six time steps into the future for both interpolations. Table 6.1 illustrates how many models that needs to be trained for each sensor combination. For each sensor combination, 8 models must be trained. That makes 96 different models to train excluding the baselines.

Table 6.1: All different models trained from one sensor combination

Sensor Combination	<example sensor combination>							
Sensor to Forecast	Pressure				Temperature			
Interpolation	10 min		1 h		10 min		1h	
Number of Forecasted Features	1	6	1	6	1	6	1	6

6.1 Baseline

To compare the results of the sensor combinations, we have chosen two baselines. The first baseline is only using the history of the single sensor that is forecast as an input to the LSTM. The second baseline is using all 24 sensors as an input to the LSTM. Since no domain knowledge is needed for doing forecast with a time series from a single sensor, this is chosen as a baseline to see if combinations of related sensors can outperform this. The reason that all 24 sensors as an input is also used as a baseline is that if this baseline model outperforms sensor combinations, these combinations will have no use.

6.2 Groups of Input

We grouped the sensors based on two different approaches: What type of sensor it is (what it is measuring), and where the sensor is located in the pipeline. By grouping in these two ways it makes it possible to say something about what has the most impact on the predictions; the sensor type or the placement of the sensor.

6.2.1 Grouped on Sensor Type

In Section 5.1, the different sensor types are explained, which are pressure (PI/PT), temperature (TI/TT), emergency safety valve (ESV), and a choke indicator (ZI). These groupings of type have been combined together to see if the inclusion or exclusion of a sensor type group will affect the forecasts. Since the choke indicator impacts the other sensor values in the system significantly, it is included in every combination. The different combinations are:

- PI/PT + ZI
- TI/TT + ZI
- ESV + ZI
- PI/PT + TI/TT + ZI
- PI/PT + ESV + ZI
- TI/TT + ESV + ZI

6.2.2 Grouped on Location

When grouping on location, inspecting the P&ID is important. By looking at where the sensors are mounted on the pipeline, groupings that are potential combinations giving good predictions can be extracted. The chosen combinations are listed below:

- Production line - Only sensors that are located in the pipeline that is transporting the product.
- Production line prior to the sensor that is predicted.
- Four first sensors in the production line, without having the predicted sensor as input.
- All sensors in the system prior to the sensor that is predicted, including sensors in other pipelines.
- Only the ZI sensor in the input.
- ZI sensor and the sensor that is predicted

Since there are multiple pipelines, it makes sense to exclude those pipelines that are not transporting the product. Both the predicted pressure sensor and temperature sensor are mounted on the production pipeline, so including only the sensors from this pipe in the input is one of the approaches when selecting sensors. This assumption has been used in the first two combinations.

The second assumption is that a failure of the pipeline will affect the sensor values regardless of whether the failure happened prior or after the predicted sensor in the pipeline. Combinations two and three are testing this assumption since they only contain sensors prior to the sensor that is predicted. However, for this to give predictive powers, the pipeline failures must occur prior to the predicted sensor.

The third approach investigates the performance of the model if the predicted sensor is not included in the input. This experiment is performed as an attempt to see if the model can predict without the predicted sensor in the input. In order for this to work, the sensors in the input must be directly correlated with the predicted sensor. This approach is present in combination three and four.

The final approach is to utilize the ZI sensor and how much it affects the rest of the sensor values. When the valve the ZI sensor is mounted on is adjusted, the values of the sensors prior to this valve is affected. This suggests that change of values in the ZI sensor can lead to a change in the predicted sensor since it is located prior to it, thus leading to a predictive

model. The fourth and fifth combination use this approach.

6.3 Data Preprocessing

We know that pressure of 0 is highly improbable, but this does not necessarily mean it can be removed. When we looked at all sensors at a point where the pressure was measured exactly 0 we found that it was rarely only one sensor that contained extreme values. In Figure 6.1 we have plotted one of these scenarios where a pressure sensor measured 0 and saw that it also was a 32% increase in another temperature sensor at the same time. Since the values in the plot are normalized, 1 and 0 mean that they are the highest and lowest measured values. The temperature was measured to be approximately 145 degrees which is a sensible value according to our domain expert. Although the pressure value is wrong and does not represent the actual pressure, it can be useful in order to help indicate anomalies. Regarding the many drops in the temperature time series, these drops are present in time series all over system meaning that they most likely are not caused by faulty sensor readings. This makes it very challenging to to foresee the anomaly, since the drop in sensor values when the system malfunctions are very similar to the ones during normal operation. In Figure 6.2 we illustrate the pipeline that has been to preprocess the data. The three first steps will be explained in the next subsections, and the forth will be elaborated in Section 6.4

6.3.1 Fetching from the API

The data was downloaded via an API offered from Cognite. This gave us the opportunity to define desired granularity and aggregation of the data. We chose to download the time series with 10 minute granularity and 1 hour granularity. The sensors in the system are reading samples at different timestamps and intervals, which makes it necessary to interpolate the data so that the timestamps of all sensors are synchronized. The interpolation method used was average interpolation where each point is the average of all the data since the previous data point, in this case 10 minutes and 1 hour. This resulted in 249360 and 42775 samples respectively. Even though interpolation was done through the API, these time series still contained some NaN values.

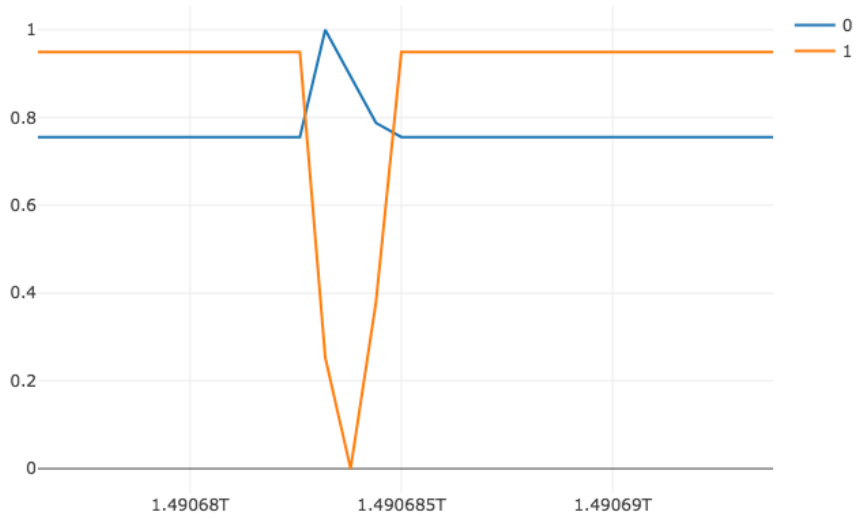


Figure 6.1: Outliers in two different sensors, the orange measures pressure while the blue measures temperature

6.3.2 Interpolation

To handle the NaN values we used Pandas[1] built-in interpolation and used the technique based on indexes because all the samples had equal distance in time to the next. Index based interpolation calculate missing values by using the numerical values of the index to weight the next and the previous value when averaging them.

6.3.3 Normalization

All values in the time series have been normalized to a value between 0 and 1 in order to improve the training of the neural networks. Feature scaling with the use of the minimum and maximum value of the time series has been used. For each time series, each value is scaled with the use of the minimum and maximum value of time series that is normalized. A challenge is present when dealing with maximum and minimum values that are very different from the mean of the time series, which is the case in many of the time series. As

[1] `https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.interpolate.html`

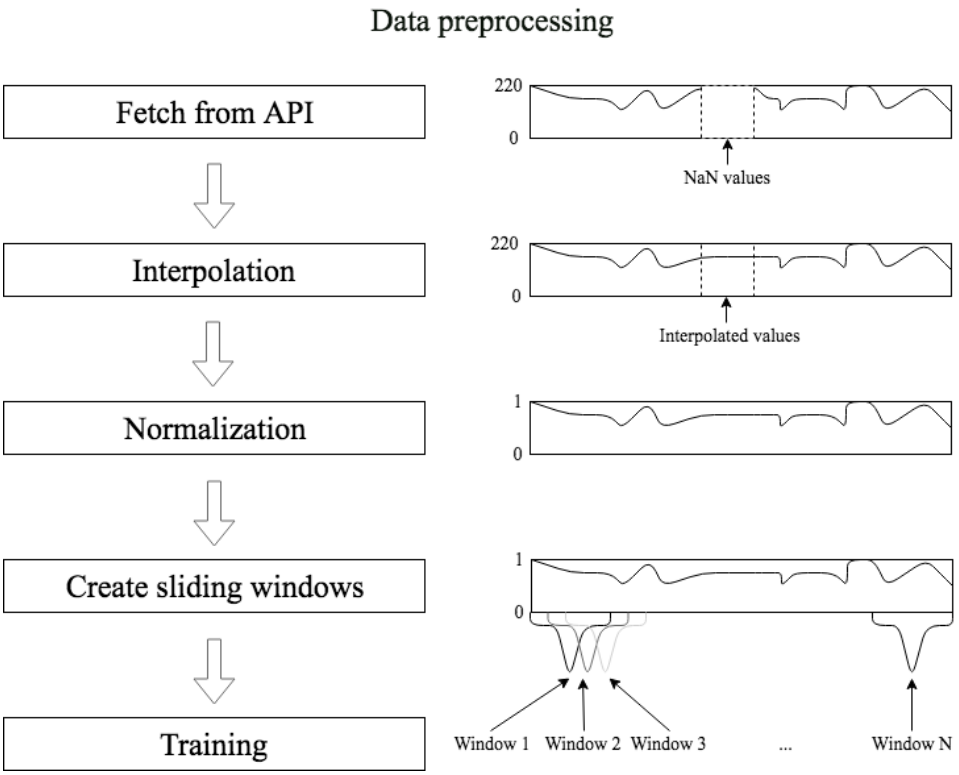


Figure 6.2: Data preprocessing step before training

a consequence, the difference between the majority of the values in the time series is very low after normalization. This can affect the predictive power of the model since it might have problems distinguishing between values of very small fractions. A solution to this is to remove all the spikes in the data, but since they are not possible to classify as faulty data, the model should preferably be able to learn and forecast them.

6.4 LSTM Input Structure

Keras running on TensorFlow[2] has been used to implement the LSTM. All time series used in the input have been transformed to a sliding window of 50, where the first 49 values are used as an input for the experiments that are predicting one time step, and the first 44 values are used as an input for the experiments that are predicting six time steps. So when

[2] `https://www.tensorflow.org/`

a single time series X is used as an input to the LSTM, the first four input windows looks like this when forecasting one time step:

$$\{[x_1, x_2, ..., x_{48}, x_{49}], [x_2, x_3, ..., x_{49}, x_{50}], [x_3, x_4, ..., x_{50}, x_{51}], [x_4, x_5, ..., x_{51}, x_{52}]\},$$

where x_1 is the value at the first time step in the time series. The values that are predicted in each window are x_{50}, x_{51}, x_{52} and x_{53}, respectively. When a single time series X is used as an input to the LSTM, the first four input windows looks like this when forecasting six time steps:

$$\{[x_1, x_2, ..., x_{43}, x_{44}], [x_2, x_3, ..., x_{44}, x_{45}], [x_3, x_4, ..., x_{45}, x_{46}], [x_4, x_5, ..., x_{47}, x_{48}]\},$$

where the forecasted values for the windows are $[x_{45}, x_{46}, ..., x_{49}, x_{50}]$, $[x_{46}, x_{47}, ..., x_{50}, x_{51}]$, $[x_{47}, x_{48}, ..., x_{51}, x_{52}]$ and $[x_{48}, x_{49}, ..., x_{52}, x_{53}]$, respectively.

In most of the experiments, multiple time series are used in the input of the LSTM. This means that an additional dimension has to be introduced in the input. Given the time series X^1, X^2, X^3 and X^4, the first four windows of the input to the LSTM looks like this when forecasting one time step:

$$\begin{aligned}
&\{[(x_1^1, x_1^2, x_1^3, x_1^4), (x_2^1, x_2^2, x_2^3, x_2^4), ..., (x_{48}^1, x_{48}^2, x_{48}^3, x_{48}^4), (x_{49}^1, x_{49}^2, x_{49}^3, x_{49}^4)],\\
&[(x_2^1, x_2^2, x_2^3, x_2^4), (x_3^1, x_3^2, x_3^3, x_3^4), ..., (x_{49}^1, x_{49}^2, x_{49}^3, x_{49}^4), (x_{50}^1, x_{50}^2, x_{50}^3, x_{50}^4)],\\
&[(x_3^1, x_3^2, x_3^3, x_3^4), (x_4^1, x_4^2, x_4^3, x_4^4), ..., (x_{50}^1, x_{50}^2, x_{50}^3, x_{50}^4), (x_{51}^1, x_{51}^2, x_{51}^3, x_{51}^4)],\\
&[(x_4^1, x_4^2, x_4^3, x_4^4), (x_5^1, x_5^2, x_5^3, x_5^4), ..., (x_{51}^1, x_{51}^2, x_{51}^3, x_{51}^4), (x_{52}^1, x_{52}^2, x_{52}^3, x_{52}^4)]\},
\end{aligned}$$

where x_1^1 is the value of the first time step in the first time series. The values that are forecasted in these four windows are $x_{50}^1, x_{51}^1, x_{52}^1$ and x_{53}^1. When forecasting six time steps, the first four windows looks like this with the time series X^1, X^2, X^3 and X^4:

$$\begin{aligned}
&\{[(x_1^1, x_1^2, x_1^3, x_1^4), (x_2^1, x_2^2, x_2^3, x_2^4), ..., (x_{43}^1, x_{43}^2, x_{43}^3, x_{43}^4), (x_{44}^1, x_{44}^2, x_{44}^3, x_{44}^4)],\\
&[(x_2^1, x_2^2, x_2^3, x_2^4), (x_3^1, x_3^2, x_3^3, x_3^4), ..., (x_{44}^1, x_{44}^2, x_{44}^3, x_{44}^4), (x_{45}^1, x_{45}^2, x_{45}^3, x_{45}^4)],\\
&[(x_3^1, x_3^2, x_3^3, x_3^4), (x_4^1, x_4^2, x_4^3, x_4^4), ..., (x_{45}^1, x_{45}^2, x_{45}^3, x_{45}^4), (x_{46}^1, x_{46}^2, x_{46}^3, x_{46}^4)],\\
&[(x_4^1, x_4^2, x_4^3, x_4^4), (x_5^1, x_5^2, x_5^3, x_5^4), ..., (x_{46}^1, x_{46}^2, x_{46}^3, x_{46}^4), (x_{47}^1, x_{47}^2, x_{47}^3, x_{47}^4)]\},
\end{aligned}$$

where the values that are foreasted are $[x_{45}^1, x_{46}^1, ..., x_{49}^1, x_{50}^1]$, $[x_{46}^1, x_{47}^1, ..., x_{50}^1, x_{51}^1]$

, $[x^1_{47}, x^1_{48}, ..., x^1_{51}, x^1_{52}]$ and $[x^1_{48}, x^1_{49}, ..., x^1_{52}, x^1_{53}]$.

6.4.1 Window Size Selection

The size of the sliding window determines how far the LSTM should look back in each input. Window size varies for the problem that is being addressed, which means that the window size should be experimented with in order to select the most appropriate size. In our experiments we decided that a window size of 49 when predicting one time step and 44 when predicting six time steps (50 in total) was the best fit. We trained an LSTM on the same time series with different window sizes and compared the MAE against each other. The result is illustrated in figure 6.3.

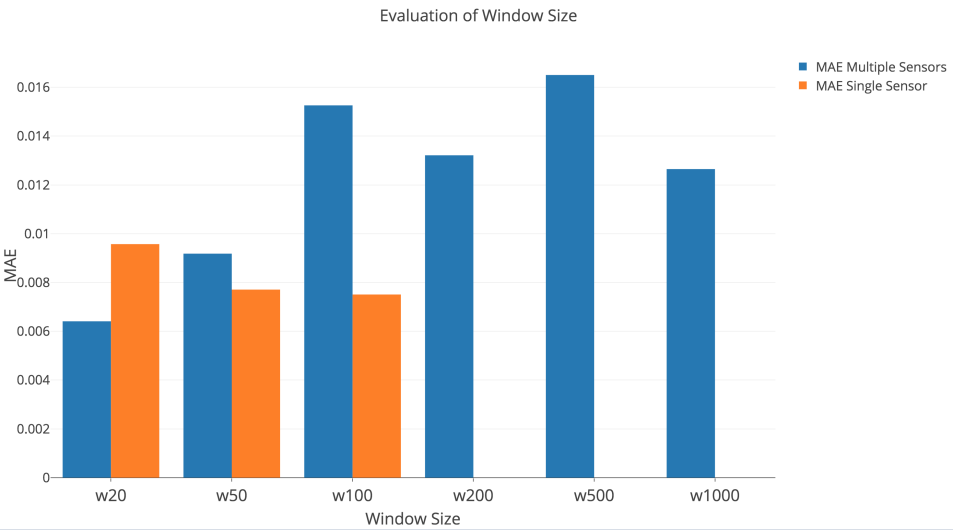


Figure 6.3: Window Size

Window sizes of 20, 50, 100, 200, 500 and 1000 have been evaluated for multiple sensors whereas window sizes of 20, 50 and 100 have been evaluated for the single sensor. For the multiple sensor, the best performing is size 20, and for the single sensor the best performing is 50. Since we want to have the same size for all the experiments, we compromised and chose a size of 49 and 44.

6.5 Stateful LSTM

We are dealing with time series, which is sequential data. The LSTM network that is implemented in Keras resets the state between batches, meaning that sequences that spans longer than one batch is not captured by the model. Since we want the model to remember the states for the whole training set, an LSTM network that resets the state after each epoch rather than each batch is implemented. This is called a stateful LSTM.

6.6 Training and Test Set

The time series were divided into training set and test sets. Presumably normal data was used as a training set, more specifically samples from index 10000 to 2000000 for the 10 minute interpolation time series and from index 1666 to index 36000 for the 1 hour interpolation time series. We created two test sets: One that contains the known anomaly and one that contains normal data. Both of these test sets are not included in the training data, so they are unseen for the model. The anomaly set contains 11667 and 1944 samples for the 10 minute and 1 hour interpolation, and the test set contains 9989 and 1667 samples for the 10 minute and 1 hour interpolation. The plots in Figures 5.2 and 5.3 show where the test sets are located in the time series. The evaluation of these test sets is presented in Chapter 7

6.7 Hyperparameters

Hyperparameters used in the LSTM network can be found in Table 6.2

Table 6.2: Hyperparameters of the LSTM network

Parameters	**Value**
Epochs	20
Batch size	32
Window size	49 and 44
Loss function	Mean absolute error
Optimizer	Adam
Prediction size	1 and 6
Dense input layer	50 neurons
LSTM layer 1	64 neurons
Dropout layer 1	0.2
Activation function LSTM layer 1	Tanh
LSTM layer 2	32 neurons
Dropout layer 2	0.2
Activation function LSTM layer 2	Tanh
Dense output layer	1 neuron and 6 neurons
Activation function Dense output layer	Linear

Chapter 7

Results

This chapter presents the results for the test set and anomaly set on the baseline models and the different sensor combinations. MAE and MSE have been used to evaluate the models. The results for the baseline are separated into two tables; one for the pressure sensor and one for the temperature sensor. For the other results, in addition to being separated into pressure and temperature, they are also divided into sensor type and location, and whether they predict one time step or six time steps.

7.1 Baselines

The results of the baselines from the LSTM experiments are presented below.

7.1.1 Baseline Pressure Sensor

Table 7.1 displays the results from the baseline for the pressure sensor that is predicted. The values for the pressure sensor normally lie between 200 and 220. For one time step, the Single Sensor baseline outperforms the All Sensors baseline on both the 10 minute interpolation and 1 hour interpolation. What is interesting to notice is that the MAE for All Sensors baseline on the anomaly set on 10 minute interpolation is slightly lower than

Table 7.1: Baseline for pressure sensor

Sensor Combination	Metric	10 min interpolation		1 hour interpolation	
		Test	Anomaly	Test	Anomaly
Single sensor	MAE	1.0650	1.4990	1.5301	1.41617
One time step	MSE	10.9232	3.6618	19.5608	10.2632
Single sensor	MAE	0.3938	1.7895	2.6129	1.7496
Six time steps	MSE	25.3231	13.4890	32.4687	32.3297
All sensors	MAE	2.0502	1.4527	3.3868	1.8306
One time step	MSE	15.1474	4.3291	25.6038	13.3423
All sensors	MAE	6.9219	3.5802	6.4499	5.6470
Six time steps	MSE	74.5154	24.0870	63.4742	55.5832

on that particular metric for the Single Sensor baseline when predicting one time step. For the prediction of six time steps, the Single Sensor has lower error than the All Sensors baseline. Another interesting observation is that there is a case where predictions of six time steps yields lower MAE than predictions of one time step. This is with the Single Sensor prediction for the 10 minute interpolation test set (0.3938 versus 1.0650). Anomaly test set error is generally lower than the test set error, especially in terms of MSE. For the Single Sensor baselines, predictions of one step has higher MAE and lower MSE than the predictions of six time steps. This means the six time steps predictions generally are closer to the curve, but have more large prediction errors.

7.1.2 Baseline Temperature Sensor

Table 7.2: Baseline for temperature sensor

Sensor Combination	Metric	10 min interpolation		1 hour interpolation	
		Test	Anomaly	Test	Anomaly
Single sensor	MAE	0.1373	0.4361	0.4227	0.41942
One time step	MSE	0.7641	0.3436	3.8166	0.56511
Single sensor	MAE	0.1586	1.3151	0.4165	0.3788
Six time steps	MSE	3.0443	3.3317	10.7320	3.8346
All sensors	MAE	1.1456	0.6961	0.7353	1.1665
One time step	MSE	2.0915	1.1456	3.2990	2.2172
All sensors	MAE	0.5743	1.7690	0.79049	0.67185
Six time steps	MSE	3.2823	4.6636	12.8533	4.3246

Table 7.2 displays the results from the baseline for the temperature sensor that is predicted. Temperature values usually lie between 107 and 110 degrees. Predictions based on values from one sensor generally have lowest loss. The only exception is the 1 hour interpolation test set Single Sensor MSE compared to the corresponding MSE in the All Sensors baseline (3.8166 versus 3.2990). Like the baselines for the forecasted pressure sensor, the MSE of the anomaly sets are generally lower than the MSE of the test sets. Even when taking into account that pressure values usually have a higher distribution, the error percentage is still lower on temperature.

7.2 Explanation of Results

In order to improve the readability of the result tables, color codes and bold text have been used. The explanation of the coding is presented below:

- **Red background:** Error is higher than both baselines.
- **Yellow background:** Error is lower than the All Sensors baseline, but higher than the Single Sensor baseline.
- **Green background:** Error is lower than both baselines.
- **Bold Text:** Error is the lowest error among the groupings on the particular evaluation dataset.

7.3 Pressure Sensor

The results for the different groupings when predicting the pressure sensor are presented in this section. The results are divided into to sections of whether the sensors have been grouped on sensor type or location. Furthermore, the results have been divided into one time step predictions and six time step predictions.

7.3.1 Grouped on Sensor Type

Table 7.3 shows the results when forecasting one time step.

Table 7.3: Pressure sensor forecast - sensor type combinations - one time step

Sensor Combination	Metric	10 min interpolation		1 hour interpolation	
		Test	Anomaly	Test	Anomaly
PI/PT + ZI	MAE	1.4045	1.4215	1.4019	2.5756
	MSE	12.5816	3.6920	18.0671	16.7748
TI/TT + ZI	MAE	**0.4828**	1.0184	2.5701	2.2536
	MSE	**10.4095**	2.6601	21.7137	15.3294
ESV + ZI	MAE	0.7817	**0.6848**	**0.8064**	**1.9886**
	MSE	11.3016	**2.03291**	**15.2780**	**13.4699**
PI/PT + TI/TT + ZI	MAE	0.9362	0.9648	5.1773	6.0932
	MSE	11.6685	2.7776	46.4438	48.4716
PI/PT + ESV + ZI	MAE	0.5228	1.0804	3.9315	4.6517
	MSE	10.9264	3.6268	32.0409	31.3457
TI/TT + ESV + ZI	MAE	1.3426	2.6716	4.1199	2.0197
	MSE	14.0528	13.5953	30.7743	16.2975

For the 10 minute interpolation the TI/TT + ZI combination gives the lowest error, however for the anomaly set the ESV + ZI combination had the lowest error. ESV beats the rest by a significant amount, even on the anomaly sets. It is also worth noticing that the PI/PT + ZI combination has the highest error for the 10 minute interpolation, but is the 2nd best combination on 1 hour interpolation.

Table 7.4 shows the results when forecasting six time steps.

Table 7.4: Pressure sensor forecast - sensor type combinations - six time steps

Sensor Combination	Metric	10 min interpolation		1 hour interpolation	
		Test	Anomaly	Test	Anomaly
PI/PT + ZI	MAE	1.0933	1.8128	1.5667	2.1402
	MSE	**25.0351**	11.3073	26.9928	36.3672
TI/TT + ZI	MAE	1.5477	2.0173	2.0107	**1.1811**
	MSE	25.9721	12.5459	24.8544	**33.6248**
ESV + ZI	MAE	**1.0748**	**0.3226**	6.2381	5.3711
	MSE	26.1081	**8.8013**	59.5566	54.0632
PI/PT + TI/TT + ZI	MAE	4.9089	3.7063	**0.9595**	3.5303
	MSE	47.1785	22.1921	22.4806	49.3099
PI/PT + ESV + ZI	MAE	2.5998	1.7485	1.4290	2.2069
	MSE	31.0006	11.73511	**22.3917**	36.2203
TI/TT + ESV + ZI	MAE	1.2025	0.7097	5.6647	6.3042
	MSE	25.9164	9.8482	51.3866	65.0793

The results are relatively equal on the 10 minute interpolation test set with the exception of PI/PT + TI/TT + ZI and PI/PT + ESV + ZI that are tad worse than the rest of the sensor combinations. On the 10 minute anomaly set, the ESV + ZI combination is the best model. The PI/PT + TI/TT + ZI combination is the best performing on the test set for 1 hour interpolation while the TI/TT + ZI is the best combination on the anomaly set. It is worth noticing how much worse the ESV + ZI model is on this interpolation compared to the 10 minute interpolation.

7.3.2 Grouped on Location

The results for the location combinations on one time step forecasts are presented in Table 7.5

Table 7.5: Pressure sensor forecast - sensor type combinations - one time step

Sensor Combination	Metric	10 min interpolation		1 hour interpolation	
		Test	Anomaly	Test	Anomaly
Production Line	MAE	2.0552	3.3871	1.2638	2.5977
	MSE	14.3792	16.2522	15.9764	17.4642
Production Line Prior	MAE	4.1138	3.7030	**0.8934**	**0.8991**
	MSE	26.3464	15.2705	**15.3584**	**10.6928**
First Four Sensors	MAE	4.7682	56.8650	15.8696	60.4363
	MSE	50.8215	5649.7413	381.4850	5450.4772
All Sensors Prior	MAE	1.5925	2.5252	1.4749	1.7246
	MSE	12.8522	8.4105	17.2915	14.6215
Only ZI	MAE	3.4458	54.3268	22.4182	58.2143
	MSE	45.9557	5441.0307	524.9211	4966.7642
ZI and Predicted Sensor	MAE	**1.4710**	**0.8289**	1.2067	1.3681
	MSE	**12.2280**	**2.3000**	17.8230	11.1364

On the 10 minute interpolation, the ZI and Predicted Sensor combination has the lowest error on both the test set and anomaly set. On the 1 hour interpolation the Production Line Prior combination has the lowest error on both evaluation sets. It is worth noticing the high error on the combinations that do not include the forecasted sensor in the input (First Four Sensors and Only ZI).

Table 7.6 shows the result for the location combinations forecasting six time steps.

Table 7.6: Pressure sensor forecast - location combinations - six time steps

Sensor Combination	Metric	10 min interpolation		1 hour interpolation	
		Test	Anomaly	Test	Anomaly
Production Line	MAE	**0.5748**	**0.4507**	1.2023	1.8419
	MSE	**24.8776**	9.3804	**22.4150**	32.8335
Production Line Prior	MAE	1.7854	1.0316	1.6526	3.2529
	MSE	27.5141	9.8673	23.3497	44.0530
First Four Sensors	MAE	7.3912	54.7019	32.0374	56.9144
	MSE	93.9835	5013.8939	1064.6584	3682.6552
All Sensors Prior	MAE	1.5909	1.3966	1.1744	1.2154
	MSE	26.0240	11.0295	23.4160	33.2533
Only ZI	MAE	35.4971	53.2883	18.6423	57.6074
	MSE	1285.2958	3131.3802	372.5788	5185.0907
ZI and Predicted Sensor	MAE	0.8852	0.6202	**0.8303**	**0.9318**
	MSE	26.7771	**8.5609**	25.1803	**32.4538**

The Production Line combination and the ZI and Predicted Sensor combination are the two best combinations. They have approximately the same error for all evaluation sets, with Production Line being slightly better for the 10 minute interpolation and ZI and Predicted Sensor being slightly better for the 1 hour interpolation. Like for the one time step predictions, the combinations without the predicted sensor included give very high error compared to the other combinations.

7.4 Temperature Sensor

This section presents the results for the different combinations when forecasting the temperature sensor. Like for the pressure sensor, the results is first divided into sensor type combinations and location combinations, and then on forecasts of one time step and six time steps.

7.4.1 Grouped on Sensor Type

The results for sensor type combinations when forecasting one time step is presented in Table 7.7.

Table 7.7: Temperature sensor forecast - sensor type combinations - one time step

Sensor Combination	Metric	10 min interpolation		1 hour interpolation	
		Test	Anomaly	Test	Anomaly
PI/PT + ZI	MAE	1.1839	2.0631	0.9073	2.4890
	MSE	2.6471	7.2200	4.2971	11.2011
TI/TT + ZI	MAE	**0.4294**	0.5068	0.4314	**0.5011**
	MSE	1.3988	3.3370	3.8336	**0.8493**
ESV + ZI	MAE	0.7666	1.4263	0.8869	2.3975
	MSE	**0.9526**	0.4928	4.1947	8.0972
PI/PT + TI/TT + ZI	MAE	0.7690	**0.4143**	1.0045	2.7755
	MSE	1.9215	**0.3276**	4.7056	13.9089
PI/PT + ESV + ZI	MAE	1.4571	0.9395	0.9056	0.5919
	MSE	3.2336	1.5475	4.1343	1.2907
TI/TT + ESV + ZI	MAE	0.7944	0.5982	**0.3716**	1.9931
	MSE	1.4944	0.4493	**3.5318**	5.7376

The results show that for the test set on 10 minute interpolation there are no combination that stands out, with the exception of the PI/PT + ZI and the PI/PT + ESV + ZI combinations that are performing significantly worse than the other combinations. For the anomaly set, the PI/PT + TI/TT + ZI combination is the combination with the lowest error. How-

ever, it is the worst performing model on the 1 hour interpolation. The combination with the lowest error on the 1 hour interpolation is TI/TT + ESV + ZI for the test set and TI/TT + ZI on the anomaly set. Only two combinations beat both the baselines.

Table 7.8 displays the results for combinations of sensor type when forecasting six time steps.

Table 7.8: Temperature sensor forecast - sensor type combinations - six time steps

Sensor Combination	**Metric**	**10 min interpolation**		**1 hour interpolation**	
		Test	**Anomaly**	**Test**	**Anomaly**
PI/PT + ZI	MAE	0.5618	1.0138	0.9320	**0.8409**
	MSE	**3.0208**	1.8852	10.1574	**4.6496**
TI/TT + ZI	MAE	**0.2822**	1.5437	**0.4801**	1.9421
	MSE	3.0954	4.3351	11.0590	9.7041
ESV + ZI	MAE	0.3252	**0.3918**	0.8464	2.0539
	MSE	3.1320	**0.4791**	**9.1851**	9.3368
PI/PT + TI/TT + ZI	MAE	0.8423	0.6405	1.2688	0.8443
	MSE	3.1815	0.9123	12.5273	4.8315
PI/PT + ESV + ZI	MAE	2.0445	0.8884	1.3383	1.4669
	MSE	10.0772	1.5351	10.9638	8.3487
TI/TT + ESV + ZI	MAE	1.4530	0.6869	1.0079	1.5575
	MSE	5.7467	0.8418	13.1890	5.7467

Like for the one time step forecasts, there are no model that stands out positively for the test set on 10 minute interpolation, but the PI/PT + ESV + ZI and the TI/TT + ESV + ZI combinations have a higher error than the other combinations. On the anomaly set, ESV + ZI has the lowest error, and all combinations except TI/TT + ZI beat both the baselines. There are no clear patterns in the 1 hour interpolation results. However, it is worth noticing that none of the combinations on the anomaly set beat any of the baselines.

7.4.2 Grouped on Location

Table 7.9 shows the results when grouping on location for one time step.

Table 7.9: Temperature sensor forecast - location combinations - one time step

Sensor Combination	Metric	10 min interpolation		1 hour interpolation	
		Test	Anomaly	Test	Anomaly
Production Line	MAE	0.7265	0.4141	1.3814	1.5551
	MSE	1.3037	0.3623	5.0910	4.3089
Production Line Prior	MAE	0.5649	1.2175	0.7213	**1.2538**
	MSE	1.4831	2.7419	4.0810	3.4819
First Four Sensors	MAE	2.3357	54.7266	1.3714	6.8991
	MSE	12.5139	5472.8043	7.9592	81.9651
All Sensors Prior	MAE	0.9920	**0.2685**	0.6235	1.8832
	MSE	1.8439	**0.1790**	3.9287	6.7945
Only ZI	MAE	1.6241	54.0104	1.9715	2.1675
	MSE	9.6958	5395.4165	8.3666	11.4134
ZI and Predicted Sensor	MAE	**0.0529**	1.2373	**0.5648**	1.3321
	MSE	**0.7374**	2.8452	**3.6152**	**2.8883**

On the test set with 10 minute interpolation, the ZI and Predicted Sensor combination has the lowest error. For the anomaly set, the All Sensors Prior combination is the best performing model. ZI and Predicted sensor is also the combination with the lowest error for the 1 hour interpolation. Very few combinations are better than both the baselines, especially for the 1 hour anomaly set where none of combinations are better than the baselines.

Results from sensor location combinations when forecasting six time steps are presented in Table 7.10.

Table 7.10: Temperature sensor forecast - location combinations - six time steps

Sensor Combination	Metric	10 min interpolation		1 hour interpolation	
		Test	Anomaly	Test	Anomaly
Production Line	MAE	0.7353	0.4801	1.3117	0.7734
	MSE	3.2249	0.6330	10.1355	5.0257
Production Line Prior	MAE	0.3673	0.5064	0.8201	1.3492
	MSE	3.3846	0.7811	9.3945	5.9235
First Four Sensors	MAE	2.2628	54.6310	2.5377	42.1000
	MSE	12.1796	5462.4680	12.3348	3176.2980
All Sensors Prior	MAE	0.7739	1.0862	1.3492	**0.6300**
	MSE	3.4280	2.0053	12.4766	**4.8555**
Only ZI	MAE	2.1950	54.5632	1.4860	1.1415
	MSE	11.8775	5455.0681	**8.7689**	12.0802
ZI and Predicted Sensor	MAE	**0.1109**	**0.4392**	**0.3478**	0.9395
	MSE	**3.0741**	**0.5692**	10.4464	4.9981

Results show that the ZI and Predicted sensor combination outperforms all other combinations on the 10 minute interpolation. For the 1 hour interpolation, the results are more ambiguous where the Only ZI and ZI and Predicted Sensor combination has the lowest error for the test set, while the All Sensors Prior combination has the lowest error for the anomaly set.

7.5 Test Set vs. Anomaly Set

Comparison of results against the baseline gives an indication of the forecasting performance of the model combinations. However, we want to know how good the model combination is on anomaly detection. By comparing the error of the anomaly set against the error of the test set we can get an indication on how well the combinations differentiates between normal and anomalous values. Figures 7.1, 7.2, 7.3, and 7.4 illustrate the differ-

ence between the MAE of the anomaly set and the MAE of the test set. Each bar in the figure represents the $MAE(anomalySet) - MAE(testSet)$ of the given interpolation and prediction size of the model combination. Since a high anomaly error and a low test error is desirable, a positive difference indicates that it can be used for anomaly detection while a value close to zero and negative may indicate the opposite. This way models that fall into the pitfall of forecasting the most recently observed value will be punished even if they have very low error on the test set.

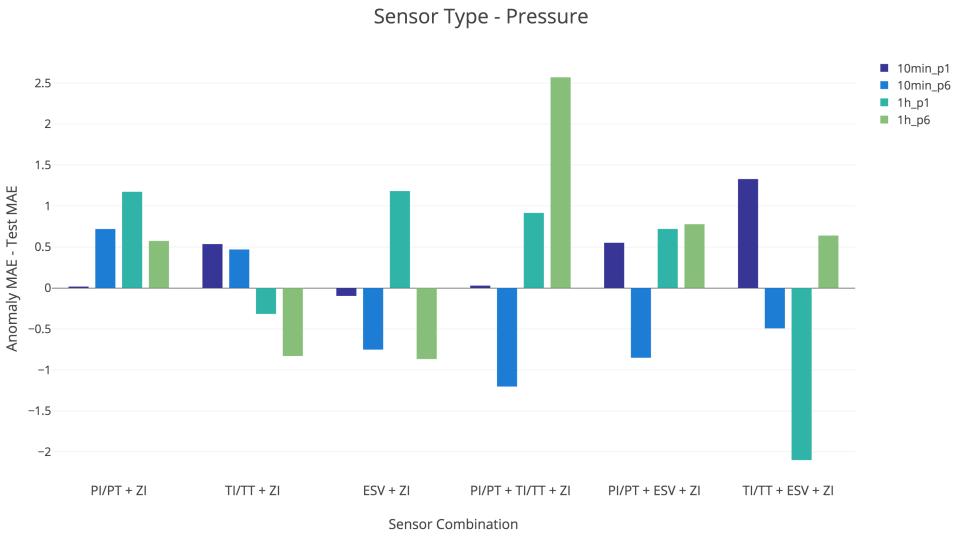


Figure 7.1: Difference of MAE between anomaly and test set - sensor type - pressure

In Figure 7.1, PI/PT + ZI is the only combination that has a positive difference on all prediction sizes and interpolations for sensor types on the pressure sensor. The TI/TT + ZI combination has a positive difference on the 10 minute interpolation, but on the 1 hour interpolation the difference is negative. The ESV + ZI combination which is the combination with the lowest error on almost every metric falls short on anomaly detection with three negative differences.

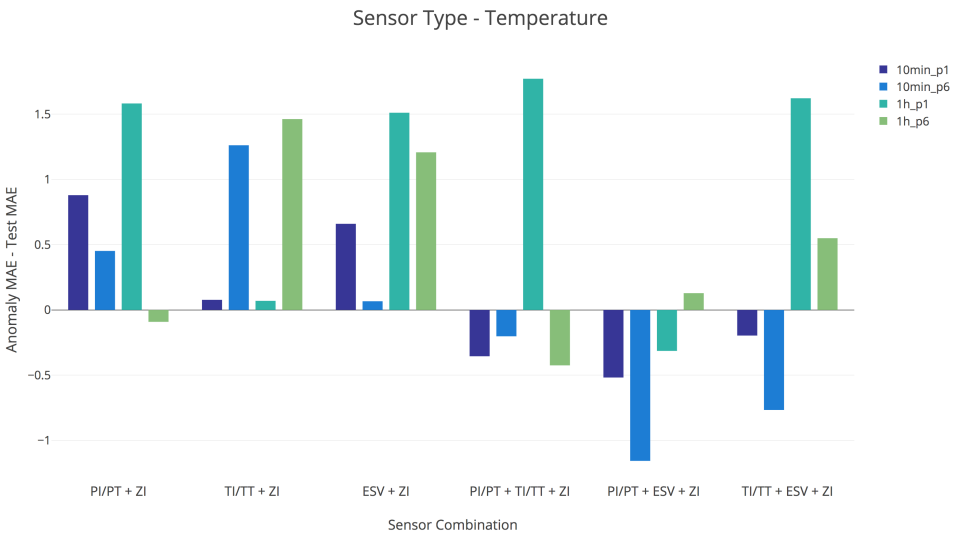


Figure 7.2: Difference of MAE between anomaly and test set - sensor type - temperature

Figure 7.2 illustrates the difference on the sensor type combinations when predicting temperature. Even though only 16/96 metrics beat the baseline (Tables 7.7 and 7.8) there are more positive differences on temperature predictions than the pressure predictions. Combinations with less sensors have a positive difference trend (PI/PT + ZI, TI/TT + ZI, and ESV + ZI) where 11/12 differences are positive, compared to 4/12 for the combinations with three types of sensors included.

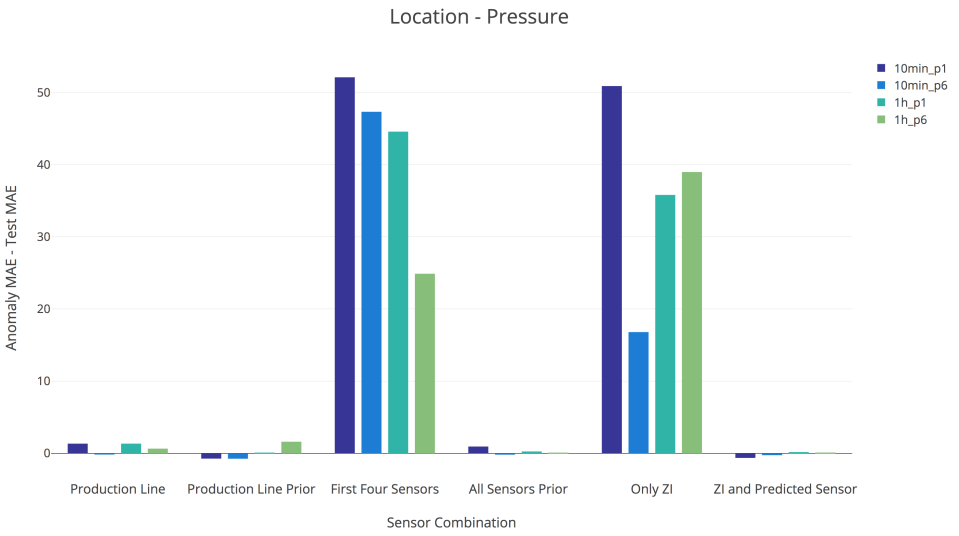


Figure 7.3: Difference of MAE between anomaly and test set - location - pressure

In Figure 7.3, the two combinations with the definitively highest positive difference are the two combinations that are excluding the predicted sensor in the input (First Four Sensors and Only ZI). Every difference metric on these two sensors are very positive. Why these combination can be used for anomaly detection will be covered in Chapter 8. Worth noticing is also the negative difference on the ZI and Predicted Sensor combination which was the best combination in terms of MAE and MSE.

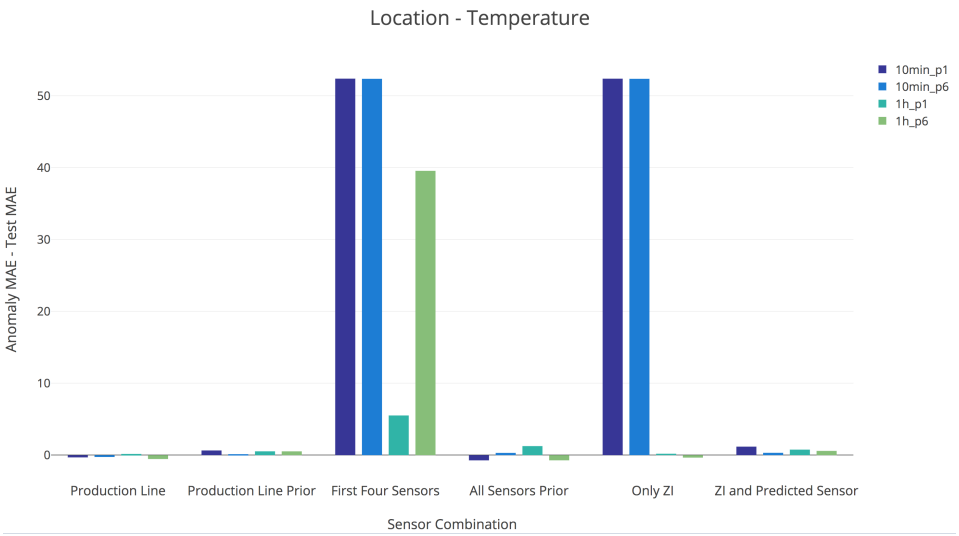


Figure 7.4: Difference of MAE between anomaly and test set - location - temperature

In Figure 7.4, as for the pressure predictions of the location combinations, the ones without the predicted sensor in the combination gives the highest positive difference. However, compared to the pressure predictions, the difference 1 hour interpolation on the Only ZI combination is dramatically lower. In Appendix C we have included a number of the plots of predictions where models have positive difference between the anomaly and the test set.

7.6 Summary of Results

- There seems to be no consistent pattern that shows that it is easier to predict 1 hour than to predict 10 minute. This was the same for both one step predictions and six step predictions. This is applicable to both temperature and pressure predictions.
- Six step predictions can sometimes be better than one step predictions, even though it theoretically should be harder to predict. Six step predictions more often beat both baselines.
- Temperature has the lowest MAE, and also lowest percentage error. The baselines errors for pressure were higher than the combination errors almost twice as many times as the baselines for temperature (69 times versus 37 times).
- MSE is much higher on pressure in general

- Only 6 out of 32 times were the lowest error of a test set from a combination of three sensor types.
- ESV + ZI and ZI + Predicted Sensor generally has very low error, ESV + ZI is somewhat lower. Both of them always have low error on anomaly test set as well.
- The sensor type that was included in combinations with the lowest error most times was the ESV combination with 16 times, closely followed by TI/TT with 14 times. PI/PT occurred only 8 times in the combinations with the lowest error.
- The two model combinations with the, by far, highest errors are the First Four Sensor combination and the Only ZI combination. These are the most interesting combinations to experiment with in regards of anomaly detection.

Chapter 8

Discussion

Selected combinations that showed interesting results from the experiments will be further discussed in the following sections. These predictions will be plotted in order to illustrate different aspects of the result.

8.1 Forecasting Power

Ten out of twelve groupings include the sensor that is forecast in its input when training the model. If the time series from this sensor is very similar to a random walk, this will lead to forecasts that to a great extent is just forecasting the value of the previous time step, since this gives a very low loss, like explained in [19]. This is illustrated in Figure 8.3 which is the one time step predictions of the pressure sensor with the TI/TT + ZI combination on 10 minute interpolation. These models have the lowest forecasting error, but no forecasting power, which renders them less useful. Not even during very anomalous behaviour does the prediction error shift significantly, as seen in Figure 8.1. These types of predictions will be named random walk forecasts for the rest of this thesis.

As mentioned in Chapter 6 and 7, two of the sensor combinations, First Four Sensors and Only ZI, do not have the predicted sensor in its input. The goal of these combinations was to see if the time series can be predicted without a direct relation between the input and output of the model. This is something that has not been tried in related work. For

this experiment to work, the sensors in the input must be correlated with the sensor that is predicted. We know for sure that the choke indicator (ZI) sensor is correlated with the rest of the sensors since adjustments to this valve affects the pressure and temperature in the system. Figure 8.4 shows a zoomed in part of the pressure sensor forecasts made from the Only ZI combination on 10 minute interpolation predicting one time step. It seems like this combination is also suffering from random walk forecasts like the combinations with the predicted sensor included in the input. However, since the forecasts are solely based on the time series from the ZI sensor, the model has possibly learned the correlation between the ZI sensor and the pressure sensor, compared to the mimicking of time series the other combinations tend to do. By looking at Figure 8.2 that illustrates the absolute forecasting error for each time step on the anomaly, we can see that the absolute error stays high after the spike, compared to the absolute error in Figure 8.1. This means that the anomaly is not only the spike itself, but also the persistent loss of pressure coming after it, which can be seen in the red area in Figure 5.2. Even though this type of combination is useful for anomaly detection, the forecasting powers on normal data is not particularly impressive. Since the value of the ZI sensor is not changing as often as the other sensor types, many predictions are the same over several time steps. This can be seen in Figure 8.5. However, these two combinations have properties that most likely are valuable for Aker BP. Some of the other LSTM models have the same performance as a rule that predicts the next value of the sensor as the previous value which make less useful. If it was not for the time horizon for this thesis, all combinations could have been retrained without the predicted sensor in the input in order to see if the forecasts could have been better while still keeping a powerful anomaly detection model.

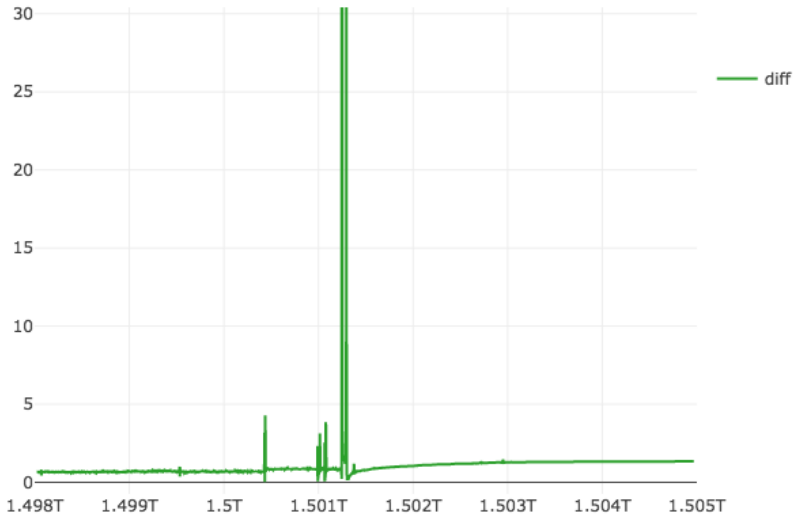


Figure 8.1: TI/TT + ZI on pressure sensor - absolute forecasting error

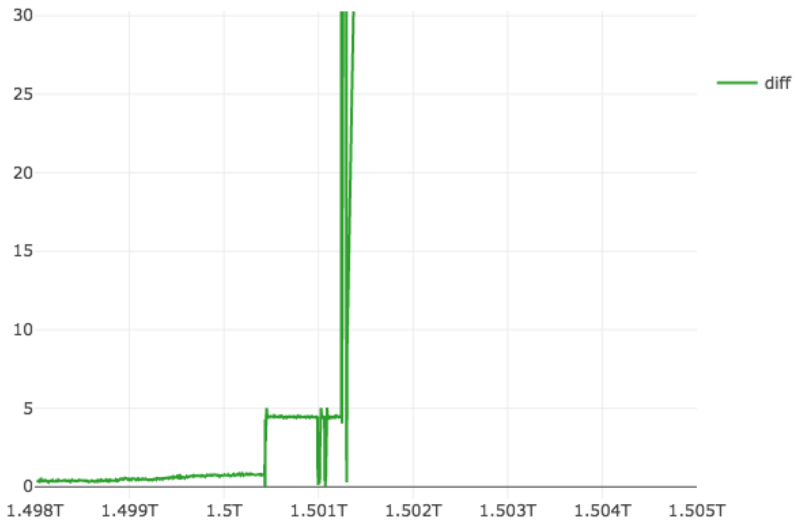


Figure 8.2: Only ZI on pressure sensor - absolute forecasting error

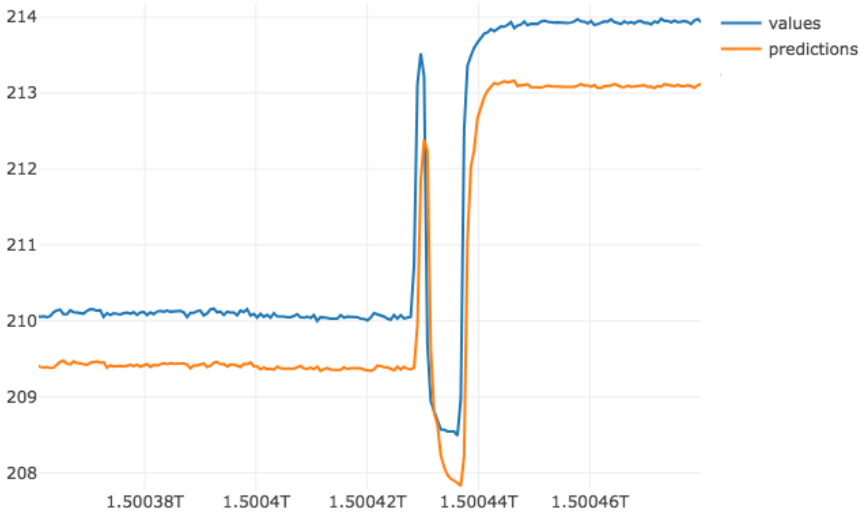


Figure 8.3: Predictions on pressure sensor with the TI/TT + ZI combination

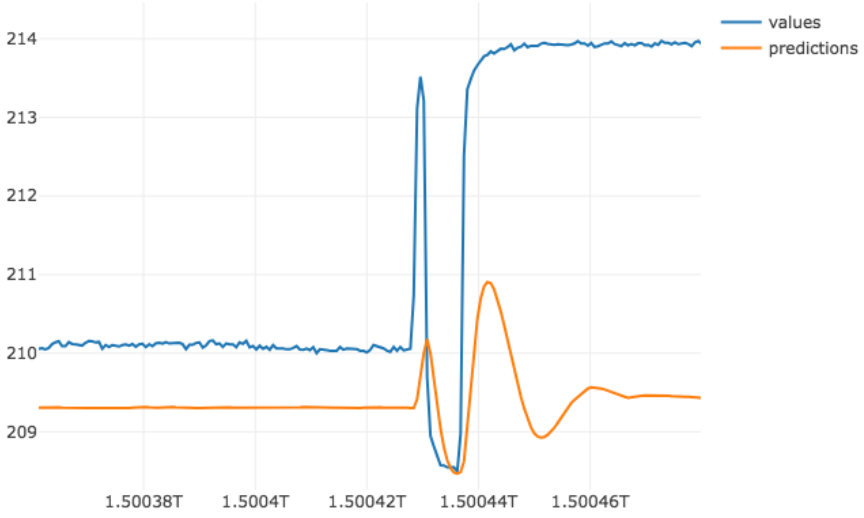


Figure 8.4: Predictions of the pressure sensor on the Only ZI combination, without predicted sensor as input

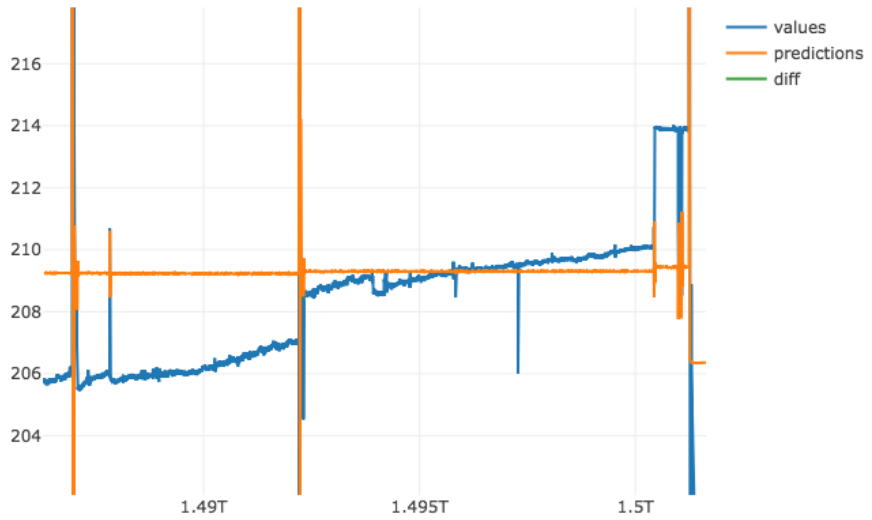


Figure 8.5: View of all predictions on the Only ZI, 10 minutes , one time step combination

8.2 Forecasting a Volatile Pressure Sensor

The models created with the different sensor combinations have generally a lower forecasting error when predicting temperature. A possible reason for this is the volatility of the pressure sensor which makes it harder to predict. By looking at the tables in Sections 7.3 and 7.4, we can see that the MSE of the pressure predictions are a lot higher than for the temperature predictions. MSE is squaring each error while MAE takes the absolute of each error. This leads to the MSE being lot higher for a prediction on a volatile time series, when the volatile pattern is not learned by the LSTM. When predicting one time step, most forecasts will be approximately the previous value of the time series, giving a significant error for each prediction when the time series is volatile.

When several time steps are predicted, models for both pressure and temperature forecasts seems to be trying to find a pattern that is matching the amplitudes of the time series. This seems to decrease MAE on some sensor combinations compared to the one step predictions which means that some of those patterns is a better match than six one time step predictions put together. Still the MSE is consistently higher and the plots tell us that

the amplitudes in the predictions often have bigger ranges than the actual values. If the model predicts six steps and there is a spike lasting one time step, the five prior predictions to this spike will all be punished for not being able to predicting it, leading to the spike error being punished a lot harder. If the spike spans over more than one time step, it will be punished even more.

Training the models with MSE or a loss function that punishes large errors even more, could possibly have given us a more aggressive model compared to training with MAE, because the punishment is so high for missing a spike. At the same time the fluctuations within smaller ranges would be punished very little compared to bigger variations. The reason for training the models with the MAE loss function was to not punish the spikes in the data, since this is a part of normal operation, but this might have backfired in the terms of the model not capturing the fluctuations of the time series.

8.3 Consistent Offset Between Forecasts and Actual Values

As mentioned in 8.1, most model combinations ended up with random walk forecasts of the time series, but why are errors so different between combinations then? The reason for this is that most of these forecasts have a constant offset either higher or lower than the actual values of the time series, like Figures 8.3, 8.8, and 8.9. This means that the error is lower if the general offset from the actual values is small. An explanation of why these offsets occur might be the dirty data in the sensor combinations due to spikes in some time series. The model tries to compensate for these spikes in the training data with an offset on all predicted values in order to minimize the loss. Cleaning the data might improve this offset problem, however using more sophisticated cleaning methods than removing NaNs is challenging, since limited or no information that explains these drops and spikes is present in the work orders. Since some of the combinations have higher offsets than others, we can assume that some sensors have a greater influence on the offset. Some combinations have even a smaller offset than the single sensor baseline, leading to the hypothesis that including some of the related sensors can positively impact the results.

If the model first gets an consistent offset from the actual values, further reduction of loss will sometimes lead to unwanted effects in the model. This effect can be seen in three of the plots in Figure 8.12, especially in the plot in the upper left corner. Here it seems

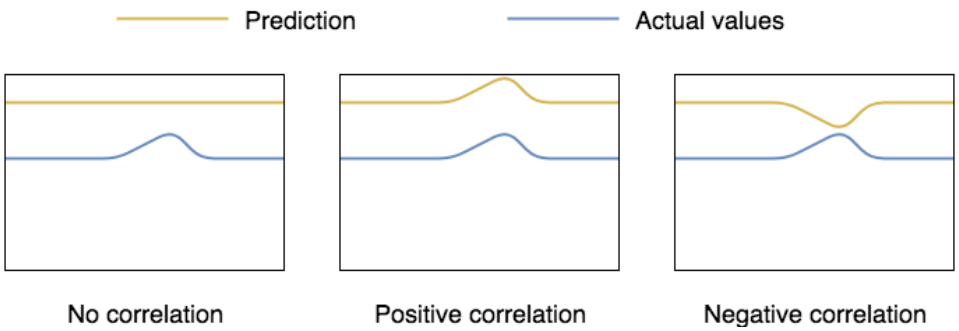


Figure 8.6: Three ways of using features from the training data with different results

like the predictions are going up when the actual values are going down and vice versa. In Figure 8.6 we try to show that if there is a constant offset, correlating with a certain feature the ”wrong” way can give lower loss. The goal of the model is to minimize loss, not to learn features we consider meaningful.

8.4 Predicting Six Time Steps

We have earlier discussed what happens to the models when they look like random walk forecasts, but can this also happen when predicting several time steps? With this assumption, a horizontal line where all six predictions are equal to the value of the most recent time step in the input, or a curve leaning towards the average value of the time series could be expected. Surprisingly, we did not find these patterns in any of the predict six time step plots.

When looking at the MAE for the 10 minute interpolation of six time step forecasts, one of the ”best performing” combinations is the ESV + ZI. This combination is only beaten by the Single Sensor baseline. When looking at a plot of the forecasts of the model, we saw that it tends to predict the same pattern for every sixth time step and move the starting point of this pattern according to recent observations. This can be seen by looking at Figure 8.7 which is a plot of the mentioned model predicting during very anomalous data, where every sixth prediction is merged into a continuous line. The plot is during a period we suspect to be a non-operating or partially-operating state. This is because it is first time in the time series we see the pressure consistently staying below 200. Again, one of the best performing sensors according to MAE, gives us the least possible indication of anomalous

behaviour. When training with mostly faulty input one might expect the model to perform worse than other models. In the ESV + ZI combination 9 out of 11 input features come from faulty ESV sensors, but this model has the lowest error of them all. Like the random walk forecasts on one time step, this combination acts in a similar way.

In Figure 8.8 we can see two figures, one where there is a constant offset from the real values and one where they overlap. While the pattern does not fit very good, it probably gives less error than a straight line with one of the recently observed values. As explained in 8.3, the most significant reason for the difference in error on the combinations is the offset.

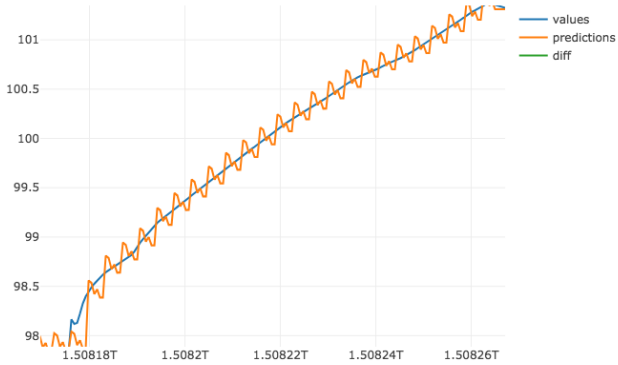
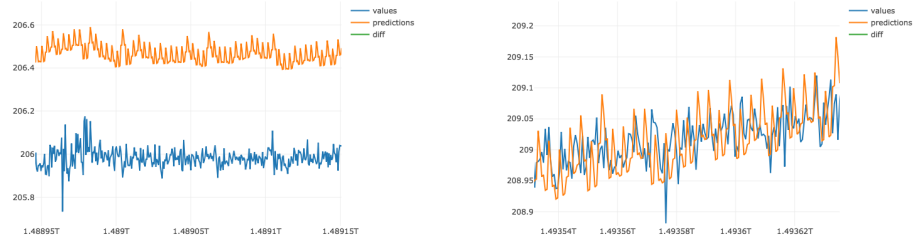


Figure 8.7: Plot of predictions on anomalous data - ESV + ZI combination - 10 minute interpolation - six time steps



(a) Without overlap

(b) With overlap

Figure 8.8: Plot of the repeating pattern on normal data - ESV + ZI combination - 10 minute interpolation - six time steps

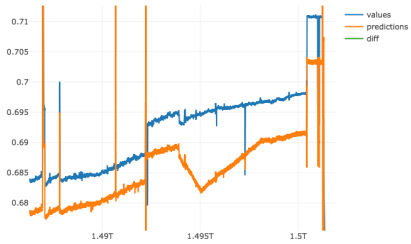
8.5 When Models get too Complex

When inspecting the plots of the different combinations we saw that some forecasts seem to have sudden deviations from the actual values. One clear example of this can be seen in Figure 8.9 where we can see two plots of predictions for six values at 10 minute interpolation for pressure. In Figure (a) we can see plots of predictions and actual values for the Production Line Prior combination, and in Figure (b) we can see the same for All Sensors Prior. Notice how there is a section around 1.495T where the predictions in plot (a) drops and then raises over a longer period, something we can not find any traces of in plot (b).

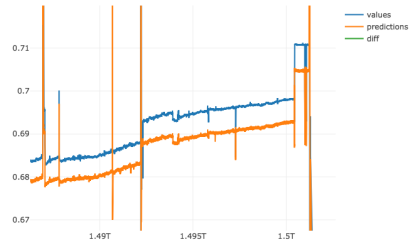
The difference between these two models is that the All Sensors Prior combination has more input sensors, specifically 2 temperature, 2 pressure and 5 ESV sensors. When plotting all the input sensors of the different combinations, we concluded that there was only one sensor that could be the cause of the drop. Because the deviational trend turns almost at exactly 1.495T, we think that sensor plotted with green in Figure 8.10(a) is the cause, because it turns at the same time. It seems to be inversely correlated to the predictions when the values of this sensor are between approximately 0.7 and 1. If this is true, why can we not see the same effect the first time the orange line is reaching 1 at the very start of the test data? This can possibly be explained by the gates in the LSTM layers. It allows a set of other values to control when the cell is going to accept a new value, and whether the value is let out. There could be one or more cells which does not allow itself to be written to or read from at the first peak, and during a relatively short period it lets the orange sensor have significant effect on the predictions before it closes again. While this does not give a better loss in the test data, it probably had a positive effect on the training data and is therefore behaving this way.

These effects can also be seen in Figure C.8 and Figure C.10 plotted in the Appendix. In general we think this is some kind of overfitting which can happen when the models that are used gets too complex and the training data is not extensively cleaned. Reducing the number of neurons or changing layer type to GRU [5] and standard RNN layer could have helped creating a less complex model.

We wanted to investigate further and check whether it was the ESV sensor plotted in green in Figure 8.10(a) that caused the drop. If our explanation is correct, setting the values of the green ESV sensor to something lower than 0.7 during the period of the drop would cause the drop effect in Figure 8.9(a) to go away without affecting the plot much in any other way. In Figure 8.11(b) we can see an altered plot of the sensor which is plotted in

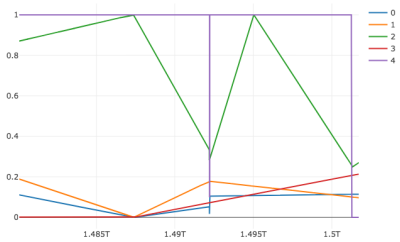


(a) Prior

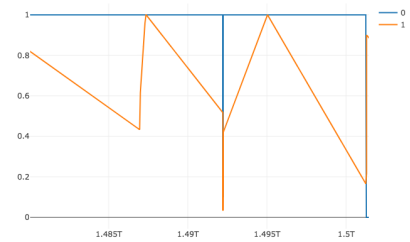


(b) Prior Production

Figure 8.9: Plot of All Sensors Prior combination and Prior Production combination



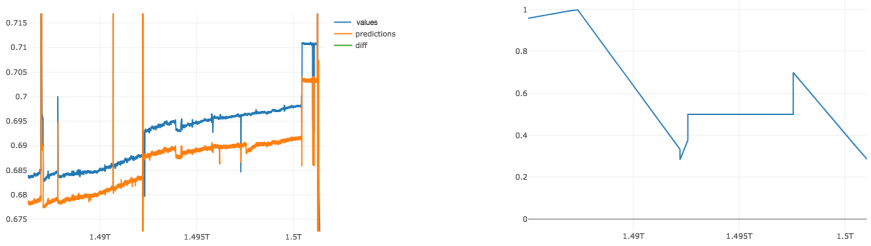
(a) ESV - All Sensors Prior



(b) ESV - Prior Production

Figure 8.10: Plots of ESV sensors in All Sensors Prior and Prior Production combinations

green in Figure 8.10(a). We set the values during the drop to be 0.5. In Figure 8.11(a) we can see the new plot of real values and predictions of the All Sensors Prior combination with the altered input, while using the same LSTM model. Where the altered values go back to the original values, a small drop can be seen in the new prediction plot because we did not adjust enough values to cover the whole drop. Now the sudden deviation has disappeared, strengthening the assumption that the specific ESV sensor was the reason for it. We think this further substantiate our claim that the model is overfitting.



(a) All Sensors Prior with altered ESV

(b) Altered ESV sensor

Figure 8.11: Predictions with altered ESV sensor and plot of the altering

8.6 More Data - Better Forecasts?

In all of our 10 minute interpolation experiments we had six times more data to train the models on, compared to the 1 hour interpolation. Theoretically this should have a generalizing effect on the models, making sure that they do not overfit, because there are so many cases of different behaviour that it should be easier to learn the valuable features. In our case the 1 hour experiments sometimes had lower error and sometimes not. There do not seem to be any clear advantage for the 10 minute predictions. On one side the 1 hour should be easier to predict because they are averages of more samples than the 10 minute data, and will fluctuate less. On the other side the 10 minute dataset is bigger, but will contain up to six times more data that could confuse the model. The ratio of good data to bad data in these models will be about the same independent of the sampling rate. In the article by Banko et al. [3] mentioned earlier, they suggest that bigger datasets can give better results even if they contain more errors, but they talk about a much bigger increase than six fold.

In Figure 8.12 we have plotted all variations of the PI/PT + TI/TI + ZI combination forecasting temperature on all the unseen data prior to the anomaly. All plots goes from 108,5 to 110 degrees on the y-axis which makes us able to compare their error by inspecting the distance to the real values here shown in blue. When comparing the two 1 hour plots to the 10 minute plots we see that the 10 minute predictions are both the best and the worst. When looking at the tables in the results for temperature predictions, six step predictions almost always perform better on 10 minute data than 1 hour data with the exception of when ESV sensors are used together with other sensors. This makes sense since we know

the ESV sensors are faulty. On one time step predictions we did not see this difference. It might be that even when the ratio of bad and good data is the same, more data gives better predictions on some combinations. However, the results are too ambiguous to conclude that this is the general case.

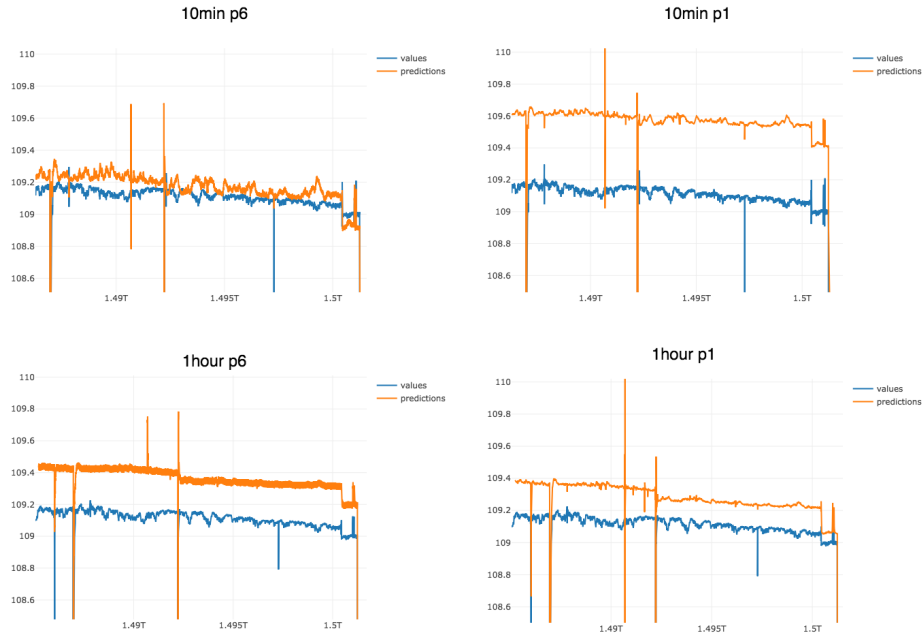


Figure 8.12: Comparison between 10 minute and 1 hour forecasts

8.7 Comparison to Related Work

Tables 8.1 and 8.2 show the MSE from two of the papers in related work, [36] and [43]. The results have been converted to MSE in order to compare the results with the MSE of our experiments. However, directly comparing the MSE does not give any particular value since the range of values are different than ours, and the data that is predicted is from other domains. Because of this, we chose to visually compare the prediction plots from the papers to our plots.

Table 8.1: Results from LSTM bug [36] predictions in MSE

Dataset	Univariate	Multivariate	Multivariate with coherence
Debian	63756	53254	44461
Mozilla	265,69	246,80	222,60
Eclipse	6369,6	5933,6	4548,1

Table 8.2: Results from bitcoin price [43] predictions in MSE

Method	MSE
LSTM	74507,16
GRU	75686,96
ARIMA-DR	65484,81

When inspecting the plots in [43], their forecasts also seem to be approximating a random walk forecast of the time series with the forecasts lagging one time step behind. Claims say that exchange rates are generally unpredictable in the short term [19], something that the plots in this paper are indicating. However, when specifically looking at their LSTM forecasts, the mimic is not as present as in their other experiments. Reasons for this might be that they do not have sufficient data size for the model to find out that there are no patterns and predicts randomly, and that they have too many other non relevant features included in their input. With over 290 features it is very probable that many of the features will not be relevant most of the time.

A partially visible random walk forecast of the time series is also present in the bug prediction paper [36]. The data they are using as input is also originating from one source, like our single sensor models, leading the LSTM in a large extent to mimic the time series that is predicted if there are no clear patterns in the data. All of their datasets have very few samples and this might be the reason that we do not see the random walk forecast effect very clearly in these data as much as in others.

The patterns of the predicted time series in [35] are temporal. This is the only paper where the LSTM apparently is able to learn the patterns, because low prediction error is present on normal patterns, and high prediction error is present all over the anomalous patterns. Even on patterns that contain spikes on the normal data, the error is low. Though, due to no obvious patterns or cycles in the sensor data that has been experimented with in this thesis, it is hard to say if the LSTM models in [35] would have been able to foresee anomalies in

our data.

8.8 LSTM and Anomaly Detection

We have to distinguish between anomaly detection and anomaly forecasting. To be able to forecast when anomalies are about to happen, there have to be some indication of this in the measured sensor values. If such indications exists, it really is the same problem, just that the model needs to detect more subtle anomalous behaviour which occurs before the more apparent anomalies. We checked a number of predictions at the exact moment of the anomalous incident to see if there some predictions were able to forecast it earlier than others, but all seemed to be rising at the exact same timestamp. This is probably because the granularity of the interpolation is too coarse to detect which sensors the anomaly reaches first.

We have established that models which do random walk forecasts give very little indication of anomalies, but how do other models perform? To do this evaluation we compare one of the models to a completely different technique executed on the same dataset. The model we picked has lower error on the test set than the anomaly set. We have conducted an experiment with an autoencoder, details can be found in Appendix B, where we wanted to see the if the reconstruction of all sensor values could give a better overview of the state of the system, and thereby detect an abnormal state at an earlier point than the LSTM. The autoencoder in our experiment does not understand the notion of time given that it is a feedforward neural network, meaning that it can not detect patterns over multiple time steps, but it can tell us if the sensor values in the input have been seen before by the model.

Figure 8.13 shows a zoomed plot of the autoencoder reconstruction error for temperature, pressure, and ZI sensors during the period before the anomaly we have been investigating. The reconstruction error is the difference between the actual values and the recreated values, and since both input and output is normalized between 0 and 1 it can be thought of as percentage error. This plot excludes the ESV sensors since we know they are faulty, but a plot including these can be found in Appendix C. To see how the reconstruction error is after the anomaly, which is the last spike in Figure 8.13, this plot is over a longer period than Figure 8.14. Figure 8.14 shows the predictions and actual values for TI/TT + ZI combination predicting temperature for 1 hour and six time steps. This is one of the combinations that show a significant difference between test error and anomaly error. The plot in Figure 8.15 shows the prediction, actual values, and the error distance between

them when the anomaly occurs.

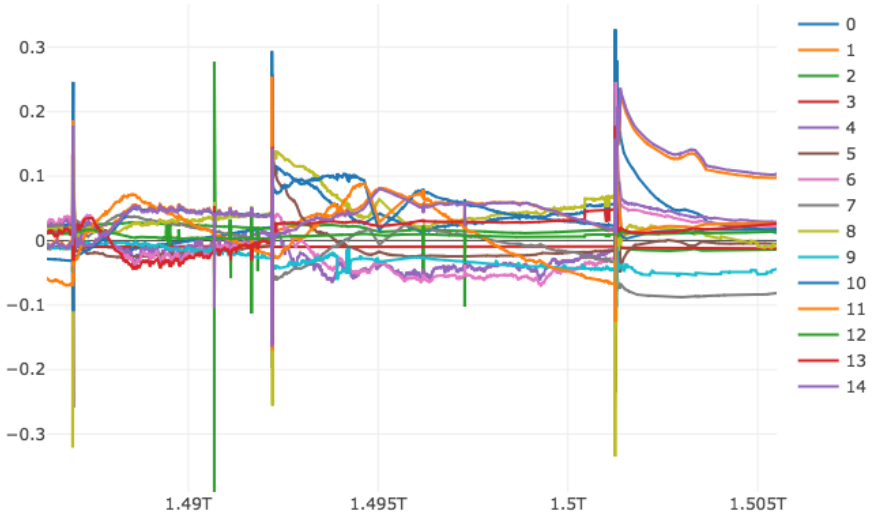


Figure 8.13: Autoencoder reconstruction error for pressure, temperature, and ZI sensors

After the anomaly, both of the techniques show a significant error. Most of the sensors have high reconstruction error during the anomaly spike after 1.5T, but only 5 sensors show a lasting high error. For the particular anomaly researched in this thesis, we think the autoencoder detects this just as well as the LSTM, with a much less complex model. There seems to be both some increases and decreases of error before the anomaly so we can not claim that the autoencoder is able to forecast the anomaly without further investigation.

Another thing that can be investigated is whether or not the autoencoder can detect the drops present in the predictions in Figure 8.14. In Figure 8.14 there are several drops in both the predictions and values. Two of them stops within the boundaries of the plot, while second and fourth goes down to about 60 and 70 degrees. The third and fifth are the only spikes where the predictions drops significantly and the actual values does not follow. The first significant error spike in Figure 8.13 corresponds to the second drop in Figure 8.14. The first predicted drop is really not expected to be seen in the reconstruction error, since the temperature only drops 0.8 degrees compared to about 40 degrees in the second. The increase in reconstruction error can also be seen in the predictions. By further inspection of the sensor that is shown in green, we found out that it is a pressure sensor which drops

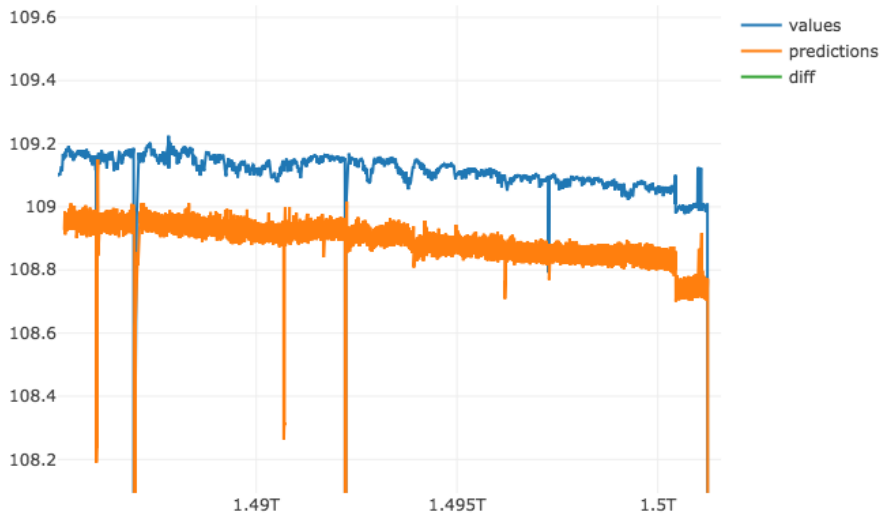


Figure 8.14: Predictions and actual values for TI/TT + ZI, 1 hour six step predictions

to 0 close to the 1.49 timestamp. This can be seen in the third drop in the predictions where the predictions drops to about 108,3 while the actual values does not change much at all. We can not say that the LSTM does anomaly detection better in this scenario because all the drops can also be seen in the autoencoder. Something that only models that predict six time steps can see, is whether the fluctuating behaviour of temperature or pressure changes. This will be seen as changes in the prediction error. After the spike in Figure 8.15 we can see the error, plotted in green, fluctuate much more heavily than seen earlier. If the values had not gone that far out of normal ranges, this might not have been detected by the autoencoder.

8.9 Importance of Domain Knowledge

Results were different for each combination which means that the sensors included in the input affected the results. Some combinations beat the Single Sensor baseline which suggest that the inclusion of related sensors can improve the forecasting accuracy. However, a portion of the combinations had a higher error than the All Sensors baseline, indicating that inclusion of everything that can potentially be related should be done with caution.

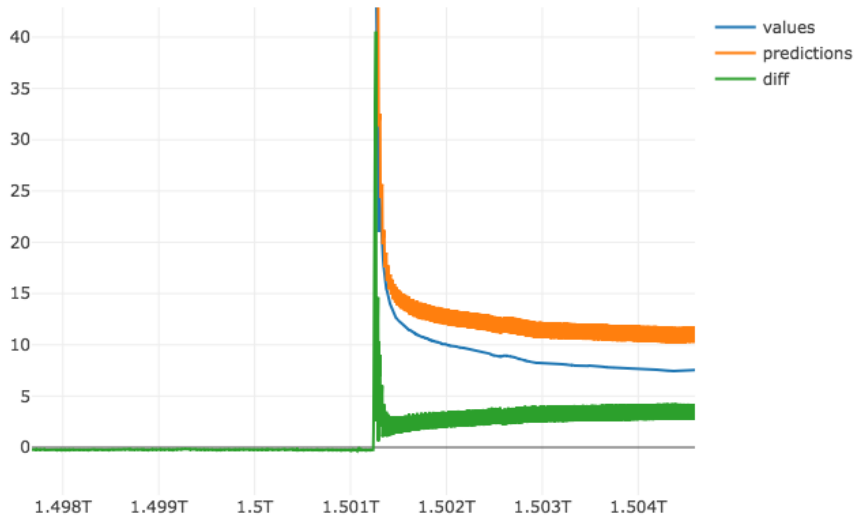


Figure 8.15: Predictions and actual values for TI/TT + ZI, 1 hour six step predictions zoomed at difference at the anomaly

In addition, when more sensors are included, there is a higher possibility for the data to be corrupt, something that occurred with the ESV sensors included in this project. On the other hand, by getting to know that the ESV time series do not illustrate the actual operation of the ESV sensors correctly, it is shown that having domain knowledge is important. If we did not figure this out, it would seem that the ESV sensors had the best impact of all the sensors.

The combinations excluding the predicted sensors were the ones with the most value. As mentioned, the input of these models needs to be related to the predicted sensor, something that would not be possible without domain knowledge.

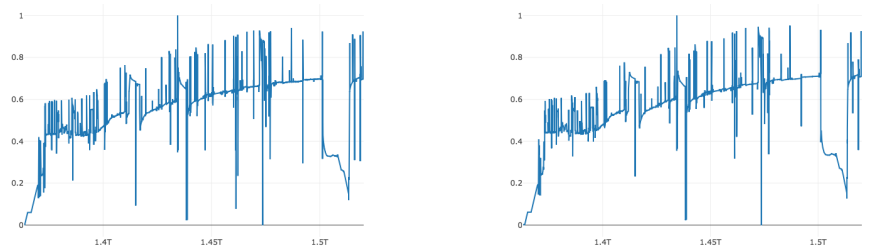
Also in general, knowledge of the domain is a big advantage. Knowing the source of anomalies, how normal operation looks like, and frequently used terms in the field can greatly improve the preprocessing of the data and evaluation of predictions.

8.10 Plausible Impacts on the Result

Like mentioned earlier, it is obvious that the faulty ESV sensors had an impact on the results. Changing the interpolation function to step interpolation rather than average interpolation on the ESV sensors could have given better results.

There are some other possible reasons that might explain the results we got. First reason is the quality of the data used as an input to the LSTM. A more extensive cleaning of the data might have improved the forecasts of the models, but due to not knowing if these outliers actually are a part of normal operation or not, lead us to not removing them. If these outliers could have been classified as anomalies, we could have converted this regression problem to a classification problem, making it easier to evaluate the models. This was something we tried, but as explained in Section 4.3.2, labeling them was not possible.

When interpolating and aggregating data to synchronize all time series on the same timestamp it is possible that certain indicators that would help identifying or predicting anomalies are lost. We reduce sampling rate to 10 minutes and 1 hour by averaging the values. It is not impossible that there could be a important change in the fluctuations which is hidden in the averaged sample. This can also happen to outliers if their duration is very short. When plotting the same sensors on 10 minute and 1 hour interpolations it seemed like the spikes were present in both plots, but a portion of the 10 minute spikes were of a larger amplitude. This can be seen in Figure 8.16 where both time series of the pressure sensor are plotted.



(a) Predicted pressure 10 minutes interpolation

(b) Predicted pressure 1 hour interpolation

Figure 8.16: Plots of predicted pressure sensor on both interpolations

Another question to ask ourselves is the predictability of the time series, whether or not if they are possible to forecast in such short terms as 10 minutes and 1 hour. The patterns

of the sensors seems to be more or less random on such fine granularity. Having a more coarser granularity might have revealed some general patterns of the time series, but the number of data samples would have decreased drastically. However, important behaviour for detecting anomalies can be smoothed out.

The final reason is tuning of the hyperparameters of the LSTM, with the tuning of the loss function being the most important aspect. If a loss function of a higher exponent was used, the model would probably have taken a bigger risk in terms of predicting fluctuations in the time series. The focus of this thesis was on the types of input data rather than the configuration of the deep learning techniques.

8.10.1 Is LSTM a Good Fit for Our Problem?

We are still not very impressed of the LSTM's performance of these types of data. The difference between the maximum and minimum value on normal operation is very low when ignoring the spikes, leading to the question if the normal operation approximates a random walk. Training an LSTM to forecasts these values in order to detect anomalies will then be a very ambitious task since it will not find any patterns of normal behaviour. Then an abnormal pattern will not be detected, and forecasting anomalies will be challenging. Detection of anomalies that manifests itself as a drop or a spike is something that can be detected by almost every technique. It would seem like good results could also be achieved with much simpler techniques, like for instance use of a weighted average of earlier sequences, where weights are determined by time lags.

The data might not be the only problem. There seems to be a tendency that the type of problems RNNs are good at solving are much more complex than what we try to do. For example speech recognition is incredibly more complex and requires one to build many levels of abstractions. In that specific case it means making out which sequences of sound that belong together and then figuring out how these make up the words. These levels of abstraction are something that we see in many of the problems that RNNs are known for performing well on.

Chapter 9

Conclusion

This chapter concludes the work done, with answers to the research questions presented in Section 1.3, and future work.

9.1 Research Questions

RQ1: *What are the main challenges of time series analysis in the oil and gas industry?*

Unexpected patterns, spikes, and drops in the time series are not documented well enough either to label it as an anomaly, clean it from training data, or consider it normal operation. This leads to input data that can be slightly dirty, which again can affect predictions negatively. Anomalies can sometimes not be reflected in the time series, since they look like spikes and drops that are considered to be normal data.

Given that the oil and gas industry is a complex field, acquiring important knowledge about it is a long process, and lack of domain knowledge can affect the results of the analysis. Given how large the industry is, multiple domain experts is needed in order to get the full picture.

We have not achieved any particularly precise forecasts on the time series. This leads us to believe that forecasting these time series precisely in as short terms as 10 minute and 1 hour time steps is very challenging due to lack of patterns on such fine granularity. However, having coarser granularity will remove important features that

are only visible in short time periods.

RQ2: *How applicable is LSTM for time series anomaly detection in sensor data?*
When comparing to another deep learning technique, the autoencoder, we did not see any significant advantage for the LSTM over the autoencoder, but our evaluation set is limited. For instance the selection of loss function, something that we could in hindsight, have experimented with a loss function punishing larger errors harder. LSTM can give extremely complex models and because of this they can become overfitted on highly irregular datasets. The more expressive the model is, the more certain you should be that the dataset do not contain faulty data.

RQ3: *How can domain knowledge of related sensors be used for anomaly detection in time series?*
The different sensor combinations ended up being both worse and better than the baselines in terms of MAE and MSE, suggesting that related sensors can affect results both positively and negatively. However, since the combinations with faulty ESV sensors had one of the lowest errors, we should be careful about drawing any conclusions on which sensors that make the most impact on the model.

By excluding the predicted sensors and using related sensors in the input forced the LSTM to learn patterns rather than mimicking the time series. In this case, domain knowledge is important since the sensors in the input of the LSTM have to be correlated to the predicted sensor for this to give good results.

9.2 Future Work

The main topic of the future work is to avoid the random walk forecasts of the time series, and we have thought of some ways to handle this:

Predicting the difference between the previous time step and the next time step, rather than the actual sensor value can show if the time series predictable or not. A problem that can arise with this approach on the temperature sensor is that the values during normal operation are very close to each other, so the model will most likely end up predicting the same difference for every input.

Two of the combinations in this thesis did not have the predicted sensor in the input, and produced some interesting results. Training all the sensor combinations again without the

predicted sensor in the input can show if this is an approach that is worth investigating further.

Training the models with a loss function that punishes large errors more can ”force” the model into learning the patterns and the fluctuations of the time series. While we tried some models with MSE without any particular results, it is possible that an experimenting with an even higher exponent would yield better results.

Bibliography

[1] Charu C. Aggarwal. *Outlier Analysis*. Springer Publishing Company, Incorporated, 2013.

[2] Charu C. Aggarwal. *Data Classification: Algorithms and Applications*. Chapman & Hall/CRC, 1st edition, 2014.

[3] Michele Banko and Eric Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th annual meeting on association for computational linguistics*, pages 26–33. Association for Computational Linguistics, 2001.

[4] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[5] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[6] Olivier Delalleau and Yoshua Bengio. Shallow vs. deep sum-product networks. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 666–674. Curran Associates, Inc., 2011.

[7] Ke-Lin Du and M. N. S. Swamy. *Fundamentals of Machine Learning*, pages 15–65. Springer London, London, 2014.

[8] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July 2011.

[9] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115, 2017.

[10] Rudolf Freund, William Wilson, and Ping Sa. *Regression Analysis*. Academic Press, 2nd edition, 2006.

[11] Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural Computation*, 12(10):2451–2471, 2000.

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

[13] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.

[14] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition, 2009.

[15] T. Hida. *Brownian Motion*, pages 44–113. Springer US, New York, NY, 1980.

[16] Schmidhuber J. Hochreiter, S. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[17] Masoud Salehi John G. Proakis. *Communication systems engineering*. 2002.

[18] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. *International Conference on Machine Learning*, 37, 2015.

[19] Lutz Kilian and Mark P. Taylor. Why is it so difficult to beat the random walk forecast of exchange rates? *Journal of International Economics*, 60(1):85 – 107, 2003. Emperical Exchange Rate Models.

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

[21] B. Ravi Kiran, Dilip Mathew Thomas, and Ranjith Parakkal. An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos. *CoRR*, abs/1801.03149, 2018.

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[23] S. Lawrence, C. L. Giles, Ah Chung Tsoi, and A. D. Back. Face recognition: a convolutional neural-network approach. *IEEE Transactions on Neural Networks*, 8(1):98–113, Jan 1997.

[24] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[25] Yann LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–50. Springer, 1998.

[26] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 609–616, New York, NY, USA, 2009. ACM.

[27] Zachary Chase Lipton. A critical review of recurrent neural networks for sequence learning. *CoRR*, abs/1506.00019, 2015.

[28] Xugang Lu, Yu Tsao, Shigeki Matsuda, and Chiori Hori. Speech enhancement based on deep denoising autoencoder. In *Interspeech*, pages 436–440, 2013.

[29] Luo, Can, Yang, Di, Huang, Jin, and Deng, Yang-Dong. Lstm-based temperature prediction for hot-axles of locomotives. *ITM Web Conf.*, 12:01013, 2017.

[30] Y. Lv, Y. Duan, W. Kang, Z. Li, and F. Y. Wang. Traffic flow prediction with big data: A deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):865–873, April 2015.

[31] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, Dec 1943.

[32] Richard Meese and Kenneth Rogoff. Empirical exchange rate models of the seventies: Do they fit out of sample? *Journal of International Economics*, 14:3–24, 1983. See also "The Failure of Empirical Exchange Rate Models: No Longer New, But Still True," Economic Policy Web Essay, September 2001. © Copyright Elsevier Science. Posted with permission of Elsevier Science. One copy may be printed for individual use only.

[33] Tomas Mikolov and Geoffrey Zweig. Context dependent recurrent neural network language model. *SLT*, 12:234–239, 2012.

[34] A. Nanduri and L. Sherry. Anomaly detection in aircraft data using recurrent neural networks (rnn). In *2016 Integrated Communications Navigation and Surveillance (ICNS)*, pages 5C2–1–5C2–8, April 2016.

[35] Gautam Shroff Puneet Agarwal Pankaj Malhotra, Lovekesh Vig. Long short term memory networks for anomaly detection in time series. *ESANN 2015*, pages 88–94, 2015.

[36] Jayadeep Pati, Krishnkant Swarnkar, Gourav Dhakad, and K. K. Shukla. Temporal modelling of bug numbers of open source software applications using lstm. In Sabu M. Thampi, Sushmita Mitra, Jayanta Mukhopadhyay, Kuan-Ching Li, Alex Pappachen James, and Stefano Berretti, editors, *Intelligent Systems Technologies and Applications*, pages 189–203, Cham, 2018. Springer International Publishing.

[37] X. Qiu, L. Zhang, Y. Ren, P. N. Suganthan, and G. Amaratunga. Ensemble deep learning for regression and time series forecasting. In *2014 IEEE Symposium on Computational Intelligence in Ensemble Learning (CIEL)*, pages 1–6, Dec 2014.

[38] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

[39] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.

[40] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, page 4. ACM, 2014.

[41] S. N. Sivanandam and S. N Deepa. *Introduction to Neural Networks Using Matlab 6.0*. Tata McGraw-Hill Education, 2006.

[42] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[43] Douglas Garcia Torres and Hongliang Qiu. Applying recurrent neural networks for multivariate time series forecasting of volatile financial data. 2018.

[44] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 1096–1103, New York, NY, USA, 2008. ACM.

[45] Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, 3 edition, 2011.

[46] Dan Xu, Elisa Ricci, Yan Yan, Jingkuan Song, and Nicu Sebe. Learning deep representations of appearance and motion for anomalous event detection. *arXiv preprint arXiv:1510.01553*, 2015.

[47] Lei Xu. Theories for unsupervised learning: Pca and its nonlinear extensions. In *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on*, volume 2, pages 1252a, 1253–1257 vol.2, Jun 1994.

[48] Amit Kumar Yadav and SS Chandel. Solar radiation prediction using artificial neural network techniques: A review. *Renewable and Sustainable Energy Reviews*, 33:772–781, 2014.

[49] Gawon Yoon. Forecasting with structural change: why is the random walk model so damned difficult to beat? *Applied Economics Letters*, 5(1):41–42, 1998.

[50] Xueyuan Zhou and Mikhail Belkin. Chapter 22 - semi-supervised learning. In Paulo S.R. Diniz, Johan A.K. Suykens, Rama Chellappa, and Sergios Theodoridis, editors, *Academic Press Library in Signal Processing: Volume 1*, volume 1 of *Academic Press Library in Signal Processing*, pages 1239 – 1269. Elsevier, 2014.

Appendix A

Paper

The paper written for ECML PKDD conference is attached on the following pages.

Impact of Domain Knowledge in Time Series Forecasting of Oil and Gas Sensor Data

Øyvind Bratvedt[1], Sigurd Grøneng[1], Trygve Karper[2], and Jon Atle Gulla[1]

[1]Department of Computer Science, Norwegian University of Science and Technology, Trondheim, Norway
oyvbr@stud.ntnu.no sigurdhg@stud.ntnu.no jon.atle.gulla@ntnu.no
[2] Cognite, Oslo, Norway
trygve.karper@cognite.com

Abstract. Deep neural networks has proven to be effective in both regression and classification of time series. Our work in this paper focuses on time series forecasting with the use of a long short-term memory (LSTM) network. The time series are sensors mounted on equipment located on an oil and gas platform in the North Sea. More specifically, the paper looks into how domain knowledge of the platform can improve the results of the forecasting. By knowing which sensors that are in proximity or in the same system of the sensor to be predicted, the time series from these sensors can be used in multivariate time series forecasting. As a baseline we used only data from the sensor we are trying to predict. The baseline has been compared to LSTM models where different groups of related sensors have been used as an input to predict the sensor used in the baseline. Some of the sensor combinations decreased mean absolute error by up to 61 percent while others increased it up to 1601 percent. At the same time some of the sensor combinations with the best forecasting accuracy showed lower error on extremely anomalous data, which suggests that it may not be appropriate for anomaly detection.

Keywords: Domain knowledge, LSTM, multivariate time series, sensor data, time series forecasting

1 Introduction

Forecasting the future values of time series can be very useful in many domains. There are many processes in the industry which depend on uncontrollable environmental parameters like weather, temperatures and more. One of many examples is power consumption which varies with days, weeks, and seasons, that makes forecasting useful for optimizing energy delivery. Forecasting time series from sensor data has also been proven to be effective in a number of scenarios in different domains like biology and weather [1, 9, 10].

This experiment features time series from a continuous oil and gas production process, fetched from an oil and gas company in Norway. In order to monitor the

production on oil and gas platforms, sensors are mounted on equipment throughout the pipeline. The current way of detecting if something is malfunctioning is to compare the values against static thresholds. Our experiment tries to incorporate domain knowledge into deep learning techniques to predict anomalies at an earlier stage, and detect the deviation of complex operational states from normal behaviour. In our scenario the most interesting use of the forecasts is to check if the production system is currently in or is transitioning into an anomalous state of operation. Detecting these situations as early as possible can be important to reduce both danger and costs. Our initial inspection of the data raised the suspicion that predicting using only one sensor's earlier history would not yield any good results because we did not find any significant patterns.

This article investigates how domain knowledge of sensors in relation to the sensor that is forecast can affect the accuracy of the forecasts, both in terms of anomaly detection and forecasting of normal operation. We chose to use long short-term memory (LSTM) networks which is known for its ability to recognize variable length dependencies in serial data. The paper is structured as follows: Section 2 presents related work, Section 3 explains the problem domain, the data source and structure of the data. Section 4 covers the theory of LSTM and predictability of time series. In Section 5, the experiments that has been performed is presented. Section 6 and 7 presents the discussion and conclusion.

2 Related Work

There are a number of previous studies that use an LSTM network to predict sequenced data.

Long Short Term Memory Networks for Anomaly Detection in Time Series [7] uses three publicly available data sets[1]. All of these data sets have patterns that can be relatively easily identified. They also use one industry data set which is not publicly available. The authors classify the industrial data set as the hardest to predict anomalies on. The goal of these experiments was to categorize anomalous behaviour, and this was done by creating an error vector which is the difference between actual and predicted values, and use this to create a Gaussian distribution. After that they calculate which threshold that gives the best precision and recall for their results. They seem to catch most anomalies using LSTM predictions, but it is not very clear how good their predictions are.

Temporal Modelling of Bug Numbers of Open Source Software Applications Using LSTM [8] tries to predict bugs using historical data. In this experiment all data that is used is originating from one univariate time series which is the number of bugs at earlier points in time. Better results was achieved when using not only the previous value to predict the next, but several time lags. In their third experiment they use partial autocorrelation to achieve somewhat better result than when using windowed data. At the same time they seem to be very much

[1] `http://www.cs.ucr.edu/~eamonn/discords/`

affected by the curve mimicking effect which will be discussed later.

Applying Recurrent Neural Networks for Multivariate Time Series Forecasting of Volatile Financial Data [11] is also related to our study. This experiment compares different types of machine learning including LSTM and GRU networks. While their data set is much smaller than ours, 1615 records, they have many more features that they use when training the model. They try to predict the price of Bitcoin using 293 different features including data from other currencies, Google Trends, and other trading data. The inclusion of all these features is interesting because one of our main questions in our experiments is whether one actually could loose precision by including irrelevant sensors in our data.

There is also another way this work is relevant for us. There are claims that financial data generally is not predictable, and at least in the short to medium term, no predictable patterns can be found in such data. What tends to happen when trying to predict these data is that the best guess is the previous value, and the prediction seems to be almost the same values, but lagging behind one time step. This is not completely the same problem as with the prediction of bugs, because they have a very clear exponential trend in the long term. Still, we can see this effect in all the three methods with best score where they use LSTM, GRU, and ARIMA with dynamic regression. This is also something that can be seen in some of our results.

The data that is dealt with in this paper has some features that makes the analysis particularly challenging. There do not seem to be any cycles or repeating patterns, and the only trend we see it that it is generally slowly increasing. These observations suggests that forecasting is more complex than predicting ECG (heartbeat activity), power consumption, or engine data like in [7]. Table 1 shows an overview of number of samples and time series in the experiments in related work, compared to the data used in this paper. This is important because more data can in many scenarios give better results. The types of input to the models are different in related work compared to our experiments. In [7] only univariate time series has been used as a feature. Bug prediction [8] is using multiple features in their models, but they are variations of the same data. Finally, [11] includes a lot of features, without knowing if they are related or not. We know that our inputs are related to each other, and they come from multiple data sources.

Table 1: Size and number of time series in related studies

Data Set	**Samples**	**Number of Time Series**
ECG - 1 [7]	3750	3
ECG - 2 [7]	5400	2
Power demand [7]	35040	1
Space shuttle [7]	5000	1
Industrial data set [7]	Unknown	Unknown
Bitcoin predicting set [11]	1615	293
Data set in this paper	190000	1-24 (depending on experiment)

3 Problem Domain

All of the time series are sensor data fetched from an oil and gas platform which collect data at regular intervals as long as they are running. They can measure temperature, pressure, and whether valves are open, closed, or in a state somewhere in between. The sensors are all related to each other in the context of being mounted on equipment from the same system. We know that equipment in this system went into a malfunctioning state, which makes it possible to evaluate how well models can detect this anomaly.

The sensors have been collecting data for about five years, also during the anomaly which happened in the middle of 2017. There are four types of sensors that have been fetched: PT/PI (pressure), TT/TI (temperature), ESV (emergency safety valve), and ZI (choke indicator). The ESV sensors measures how much the emergency safety valve must be opened to release pressure from the pipeline. The ZI sensor is mounted on the only valve that can be manually adjusted in order to manipulate the flow of product through the system. This makes it a very important sensor since adjustments performed on this valve will backpropagate into the system, changing the values in the other sensors. Figure 1 shows where sensors are located in relation to each other in the production pipeline. The sensors that are not shown in the figure are mounted on parts that are connected to the production pipeline.

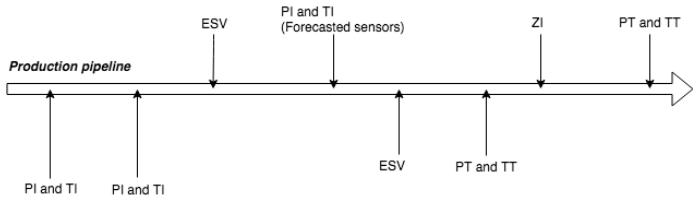
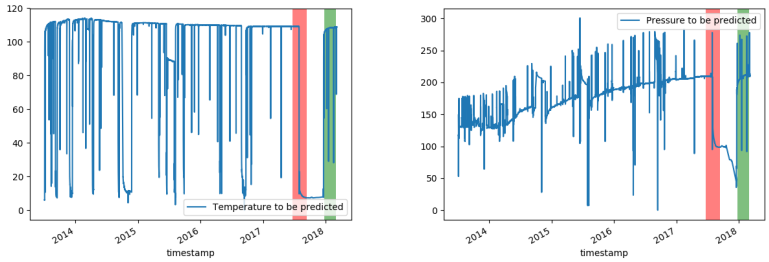


Fig. 1: Sensor network - production pipeline

In Figure 2 we can see the full plots of both the sensors that has been predicted in our experiments. The section marked with red is where the malfunctioning happens, something that manifests itself by a huge drop in values in both sensors. The section marked with green is where the operation is back to normal, and also the values that was used for evaluation.



(a) Temperature (b) Pressure

Fig. 2: Plot of the sensors we are trying to predict

3.1 Data Set

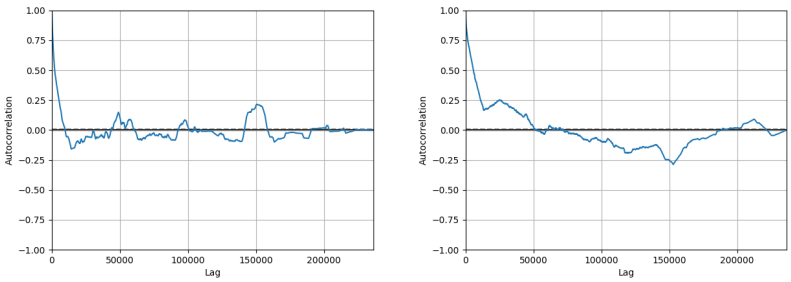
The data set used consisted of CSV files from 24 different sensors all marked with synchronized timestamps, so they all started and ended at the same time. Exactly how many sensors of each type there is can be seen in Table 2. Most sensors are sampling every minute and the data we use is the average of all the available samples from every 10 minutes. This gives a total of 249,362 samples over a period of almost five years.

Table 2: Sensors fetched from the platform

Sensor Type	**Unit of Measurement**	**Number of Sensors**
Pressure	Psi	7
Temperature	Degrees Celsius	7
ESV	Value ranging from 0-1 of how open the valve is	9
Choke indicator	Percentage of how open the valve is	1

A method to detect patterns or checking the correlation of the data is by inspecting the autocorrelation plot of the time series. This shows how correlated

values are over all possible lags. Figure 3 shows the autocorrelation plots of Figure 2. A slowly decreasing linear curve means there is a strong autocorrelation. For at least the first 10,000 time lags this seems to be the case for both time series.



(a) Temperature autocorrelation (b) Pressure autocorrelation

Fig. 3: Autocorrelation of sensors

4 Time Series Analysis

This section presents theory that is relevant for our experiments.

4.1 LSTM

LSTM (long short-term memory) is a recurrent neural network first introduced by Hochreiter and Schmidhuber in 1997 [3]. The purpose of the LSTM was to handle the vanishing gradient problem of recurrent neural networks (RNN). Even though RNNs are constructed to remember features over short time periods, this information vanishes over time as new information is fed to the network. With the introduction of LSTM, the network can remember up to 1000 steps back in time [3]. The LSTM contains information in memory cells with gates deciding what information to store and what information to output. Gers, Schmidhuber, and Cummins, 2000 [2], introduced forget gates to the cells in order to reset the state of the network. This is to prevent the network to eventually break down because of its growing state. The forget gate removes the information that is not needed anymore by the LSTM, leading to a better performing network. Out of the recurrent neural network architectures, LSTM is the one that generally has the best performance on machine learning related problems [4].

4.2 Random Walks and Predictability

In financial theory a random walk is a non-stationary time series where each step seems to be random step away from the previous value. This makes it seemingly impossible to predict and that is the case of many types of financial data. When a time series approximates a random walk, it is very hard to outperform a rule that is just guessing the previous value [6, 12]. The reason that the previous value is often a good guess, is because there is a correlation between earlier time steps which means we know the next value will not be very far away from the previous. Sometimes time series which is similar to random walk at short term, can be proven to be predictable at longer periods [5].

This problem is relevant for our work because it is not only in financial data this problem is seen. In the work done by Torres et al. [11] we can see this pattern in all of their results, including the predictions which do not use deep learning. Still, the previous guess rule may be outperformed when using other correlated data. There may very well be many correlations with other available data sources which makes us able to give better predictions than the previous value guess, but this may require knowledge of the domain.

5 Experiments

The LSTM models were all trained to forecast two sensors: One temperature sensor and one pressure sensor, located at the same place in the oil pipeline. The different approaches for dividing sensors into multivariate time series to use as an input in the LSTM is explained in the next section. These multivariate inputs are compared against the univariate input of the sensor that is forecast.

5.1 Groups of Input

We divided the sensors in two different ways: First based on what type of sensor it is, more specifically what the sensor is measuring, and the second on the location of the sensor in the pipeline.

Grouped on Sensor Type As explained in Section 3, the data consisted of four different types of sensors; pressure (PI/PT), temperature (TI/TT), emergency safety valve (ESV), and a choke indicator (ZI). The choke indicator is a major influencer of the operation in our scenario, and therefore included in all training sets. All possible combinations of these three types has been used as input to the LSTM and produced variable results. The different combinations are:

- PI/PT + ZI
- TI/TT + ZI
- ESV + ZI
- PI/PT + TI/TT + ZI
- PI/PT + ESV + ZI
- TI/TT + ESV + ZI

Grouped on Location Another way of grouping the time series is by its location in the pipeline. Six different groupings have been tested:

- Production line - Only sensors that is located in the pipeline that is transporting the product.
- Production line prior to the sensor that is predicted.
- Four first sensors in the production line, without having the predicted sensor as input.
- All sensors in the system prior to the sensor that is predicted, including sensors in other pipelines.
- Only the ZI sensor in the input.
- ZI sensor and the sensor that is predicted

5.2 LSTM Configuration and Training Set

The LSTM network is implemented using Keras[2]. In all our predictions we split the training data into sliding windows of size 50, where 49 of those values are used to predict the 50th value: $x_{t+1} = P([x_{t-48}, x_{t-47}, ...x_{t-1}, x_t])$, where x_t is the time series value at timestamp t, which is the timestamp prior to the predicted value. Since time series are sequential, stateful LSTM has been used. This means that the internal state is not reset between batches, but rather after each epoch. By keeping the same state throughout the whole epoch, you make sure that the network can capture patterns in the data that are spanning over multiple batches during training. The full model configuration can be found in Table 3

Table 3: Hyperparameters of the LSTM network

Parameters	**Value**
Epochs	20
Batch size	32
Window size	49
Activation	Linear
Optimizer	ADAM
Prediction size	1
Dense	50 neurons
LSTM layer 1	64 neurons
Dropout	0.2
LSTM layer 2	32 neurons

5.3 Evaluation of Models

To evaluate the performance of the models, two test sets were extracted from the data set. One anomaly set that contains data prior and during the malfunction,

[2] `https://keras.io/`

and one test set that only contains normal operation. The purpose of the anomaly set is to evaluate whether or not the models are able to forecast the anomalous behaviour, whereas the test set is to evaluate the forecasting of normal behaviour. For the test set, mean absolute error (MAE) and mean squared error (MSE) have been used to evaluate the performance of the model. Using these metrics when evaluating the anomaly set would not work, since a high error in the forecast does not necessarily mean that it is a poor performing model. More specifically, it can mean that the model expects normal behaviour and because of that has a large prediction error when the anomaly occurs, leading to a high MAE and MSE. By looking at the plots of the forecasts together with the actual values, it can be possible to see if the model detected the anomaly.

Baseline We have chosen two baselines to compare the different sensor combinations against. One is the forecast using only the predicted sensor's history, and the other is using all sensors. The single sensor is selected as a baseline since no domain knowledge is needed to use it as an input, and therefore it can serve as a comparison to the input combinations that utilizes the related sensors. Using all sensors in the input serves also as a baseline since if this performs better than the other combinations listed in 5.1, there will be no need for these groupings. The evaluation of these models are found in Table 4

Table 4: Evaluation of baseline

	Pressure		**Temperature**	
Sensor	**MAE**	**MSE**	**MAE**	**MSE**
Single Sensor	1.0650	10.9232	0.1373	0.7641
All Sensors	2.0502	15.1474	1.1456	2.0915

Results show that using a single sensor as an input outperforms using all sensors as an input with both pressure and temperature sensor. An explanation for this might be that the model is having trouble finding the relevant features to use when using all sensors as an input, making the forecasts less accurate. This suggests that one should be careful when adding sensor data to the training set just because one believes they are related.

Sensor Type The results of combining inputs on sensor type is presented in Table 5

Table 5: Evaluation of models grouped on sensor type

	Pressure		**Temperature**	
Sensor	**MAE**	**MSE**	**MAE**	**MSE**
PI/PT + ZI	1.4045	12.5816	1.1839	2.6471
TI/TT + ZI	**0.4828**	**10.4095**	**0.4294**	1.3988
ESV + ZI	0.7817	11.3016	0.7666	**0.9526**
PI/PT + TI/TT + ZI	0.9362	11.6685	0.7690	1.9215
PI/PT + ESV + ZI	0.5228	10.9264	1.4571	3.2336
TI/TT + ESV + ZI	1.3426	14.0528	0.7944	1.4944

Judging by the MAE and MSE of the different models, temperature in combination with the choke indicator (TI/TT + ZI) provides the best result for the pressure sensor. For the temperature sensor, temperature in combination with the choke indicator also performs well, but has a slightly higher MSE than ESV + ZI. For the pressure sensor, all models performed better than the baseline that is using all sensors in the input. The TI/TT + ZI combination also performed better than the single sensor baseline in terms of both MAE and MSE. In terms of MAE, ESV + ZI and PI/PT + ESV + ZI performed better than the single sensor baseline. For the temperature sensor, the single sensor baseline outperforms all combinations. However, some of the combinations are better than the all sensors baseline.

Location The results of combining inputs on location is presented in Table 6

Table 6: Evaluation of models grouped on location

	Pressure		**Temperature**	
Sensor	**MAE**	**MSE**	**MAE**	**MSE**
Production Line	2.0552	14.3792	0.7265	1.3037
Production Line Prior	4.1138	26.3464	0.5649	1.4831
First Four Sensors	4.7682	50.8215	2.3357	12.5139
All Sensors Prior	1.5925	12.8522	0.9920	1.8439
Only ZI	3.4458	45.9557	1.6241	9.6958
ZI and Predicted Sensor	**1.4710**	**12.2280**	**0.0529**	**0.7374**

Results show that these groupings generally are worse than grouping the input on sensor type. The exception is the combination of ZI and the predicted sensor as input. On the pressure sensor, it performs approximately the same as the sensors that are grouped on sensor type. On the temperature sensor it outperforms the single sensor baseline and has an improvement of 61 percent. The worst performing models are First Four Sensors and Only ZI. This is because

these two models does not include the predicted sensor in the input. Even though they have poor prediction error, they potentially are the best models to use for anomaly detection. Why that is the case will be covered in Section 6.

Plots We have included some plots where the predictions are concatenated into a continuous graph. These are included to corroborate certain points in the discussion. Figures 4 and 5 illustrate the prediction error difference for each forecast on the anomaly for the TI/TT + ZI combination and the Only ZI combination when forecasting the pressure sensor. Figures 6 and 7 show the forecasts of the TI/TT + ZI combination and the Only ZI combination on a typical spike in the test set of the pressure sensor.

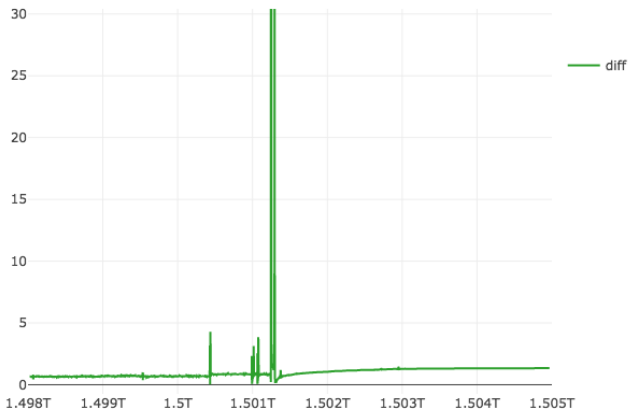


Fig. 4: Temperature and ZI sensors - forecasting error

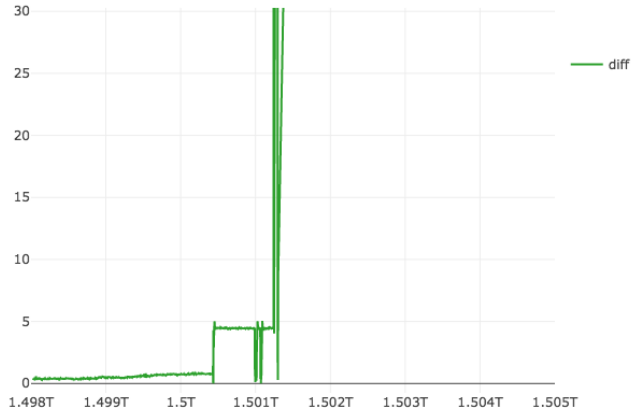


Fig. 5: Only ZI sensor - forecasting error

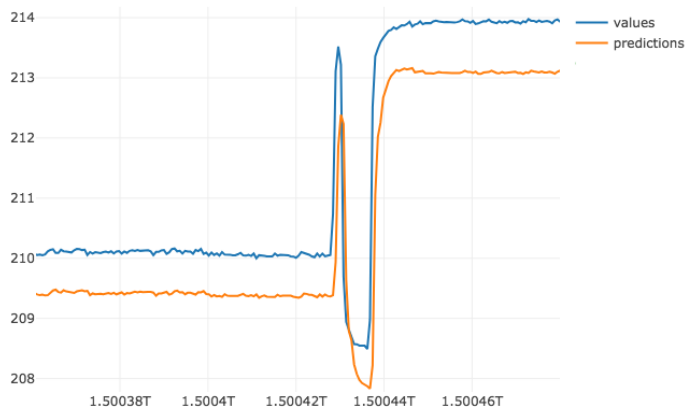


Fig. 6: Temperature, ZI and predicted pressure sensor as input

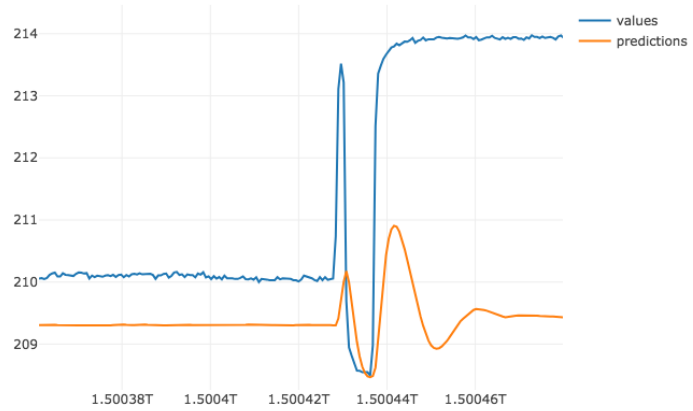


Fig. 7: Predictions using only ZI, without predicted sensor as input

6 Discussion

Generally, the models have a better prediction accuracy on the temperature sensor than the pressure sensor. The reason for this is may be that the pressure sensor is more volatile and thus more difficult to predict. This also explains why the MSE is generally a lot higher for the pressure predictions than for the temperature predictions. MSE punishes large prediction errors, giving a volatile sensor a higher MSE score than a non-volatile sensor if the frequency of the time series fails to be captured by the LSTM. Most of the forecast will be based on the previous value of the time series, giving a significant error for each forecast on a volatile time series. In addition, including temperature sensors in the input is better than pressure and emergency safety valve sensors. This is also most likely because of the volatility of pressure and emergency safety valve sensors, making it difficult for the model to find a pattern. Something that is common for the bad performing models is that all predictions either are shifted above or beneath the actual values through the entire time series. A reason for this might be that among all the 24 different time series there are numerous spikes and there are also areas with anomalous data, even in the sensors we are trying to predict. The offset could be an attempt at compensating for these small periods with suddenly very high loss. Extensive cleaning of the data might fix this problem, but also there might not exist data on what actually happened during the spikes and even if there was, identifying the parts and removing them would require intricate domain knowledge and a lot of manual work.

Ten out of twelve groupings includes the sensor that is forecast in its input when training the model. This leads to forecasts that to a great extent is just guessing the value of the last time step, like seen in Figure 6. This gives the lowest loss, but still no real insight, which renders them less useful. Not even during very anomalous behaviour does the prediction error shift significantly, as seen in Figure 4. In order to avoid this, we excluded the predicted sensor from the input, something that has not been tried in the other related work. However, in order for this to give satisfactory results, the sensors in the input have to be correlated with the predicted sensor. An example of this is the choke indicator (ZI) that is the only valve in the system that is adjustable, which will affect the values of the other sensors. When looking closely at Figure 7, we see that the prediction using only the ZI sensor seems to be suffering from the same problem, but because because predictions are based solely on the ZI sensor, the network has actually learned a correlation instead of just mimicking the curve. When looking at the difference in Figure 5, we can see that it is not the spike that is an anomaly, but the elevated level of pressure after it. One could say the given a certain ZI value, we usually see pressure values like the ones given in the predictions. At the same time this seems to limit the models ability to predict well on other areas because in the first half of the data set it basically predicts the same value.

The results on the financial data predictions [11] is not directly comparable because their range of values in the time series is much larger than ours, which gives a higher loss. Like in our plots, we still see that the predictions seems to be approximating random walk forecasts which makes the predictions lagging one time step behind. The similarity is not as clear in their LSTM predictions, and we can think of two possible explanations that might cause this: One is that they might not have enough data to make the model realize there are no patterns, where we have about 100 times more data which might be enough for that. The other possible explanation is that their other features is causing LSTM to predict this way because of actual learned patterns.

The first explanation might also be applicable to the bug prediction series [8], which is similar to our single sensor predictions because they only use historical data from one source to predict.

In [7], the LSTM is able to learn the highly temporal patterns in the time series they are forecasting. Since the time series used in this paper does not seem to have any obvious patterns, it difficult to say if their models would have detected the anomalies in our data.

7 Conclusion

In this paper, we have experimented with how different combinations of inputs to an LSTM network affects the forecasts. Most models ended up mimicking the time series that was predicted by guessing the previous value. Still, the different combinations of input produced different results which means the groupings had an impact on the predictions. Some of the combinations had lower prediction

error than the single sensor baseline, but there were also combinations that performed significantly worse. This suggests that domain knowledge can impact the results of the forecasts. However, since some of the results worsened the predictions, and because the all sensors baseline was one of the bad performing models, one should not uncritically include everything that can be related.

Our experiments also indicates that when using next-value forecasts for anomaly detection, one should verify that the model is not just mimicking previous values because that will give no real indication of anomalous behaviour. This use case is really where domain knowledge can give much better results. By only feeding data that is known to be correlated with the help of domain knowledge, we have demonstrated that we can tell whether a spike or elevation actually is expected or not. Finding the correct features to include can be very cumbersome and it can lead to both positive and negative impact, it all depends on how related the features are. Also, lower loss does not necessarily result in a more useful model.

7.1 Future Work

In order to avoid the mimicking of time series, training the model to predict the difference from the value in the previous time step instead of the actual value itself, might reveal if the time series is possible to forecast or not. Also, trying to forecast multiple time steps ahead in time can also avoid the one lag issue. This can be done by including the forecast from the previous time step as an input to the forecast of the next time step. By forecasting multiple time steps ahead, the forecasts will become less accurate, but it can give an indication of how far ahead in time the model can foresee, and also provide better applicability of the model.

Bibliography

[1] Chang, F.J., Chiang, Y.M., Tsai, M.J., Shieh, M.C., Hsu, K.L., Sorooshian, S.: Watershed rainfall forecasting using neuro-fuzzy networks with the assimilation of multi-sensor information. Journal of Hydrology 508, 374 – 384 (2014)

[2] Gers, F.A., Schmidhuber, J., Cummins, F.: Learning to forget: Continual prediction with lstm. Neural Computation 12(10), 2451–2471 (2000)

[3] Hochreiter, S., S.J.: Long short-term memory. Neural Computation 9(8), 1735–1780 (1997)

[4] Jozefowicz, R., Zaremba, W., Sutskever, I.: An empirical exploration of recurrent network architectures. International Conference on Machine Learning 37 (2015)

[5] Kilian, L., Taylor, M.P.: Why is it so difficult to beat the random walk forecast of exchange rates? Journal of International Economics 60(1), 85 – 107 (2003), http://www.sciencedirect.com/science/article/pii/S0022199602000600, emperical Exchange Rate Models

[6] Meese, R., Rogoff, K.: Empirical exchange rate models of the seventies: Do they fit out of sample? Journal of International Economics 14, 3–24 (1983), see also "The Failure of Empirical Exchange Rate Models: No Longer New, But Still True," Economic Policy Web Essay, September 2001. © Copyright Elsevier Science. Posted with permission of Elsevier Science. One copy may be printed for individual use only.

[7] Pankaj Malhotra, Lovekesh Vig, G.S.P.A.: Long short term memory networks for anomaly detection in time series. ESANN 2015 pp. 88–94 (2015)

[8] Pati, J., Swarnkar, K., Dhakad, G., Shukla, K.K.: Temporal modelling of bug numbers of open source software applications using lstm. In: Thampi, S.M., Mitra, S., Mukhopadhyay, J., Li, K.C., James, A.P., Berretti, S. (eds.) Intelligent Systems Technologies and Applications. pp. 189–203. Springer International Publishing, Cham (2018)

[9] Seal, V., Raha, A., Maity, S., Mitra, S.K., Mukherjee, A., Kanti Naskar, M.: A Simple Flood Forecasting Scheme Using Wireless Sensor Networks. ArXiv e-prints (Mar 2012)

[10] Sparacino, G., Zanderigo, F., Corazza, S., Maran, A., Facchinetti, A., Cobelli, C.: Glucose concentration can be predicted ahead in time from continuous glucose monitoring sensor time-series. IEEE Transactions on Biomedical Engineering 54(5), 931–937 (May 2007)

[11] Torres, D.G., Qiu, H.: Applying recurrent neural networks for multivariate time series forecasting of volatile financial data (2018)

[12] Yoon, G.: Forecasting with structural change: why is the random walk model so damned difficult to beat? Applied Economics Letters 5(1), 41–42 (1998), https://doi.org/10.1080/758540124

Appendix B

Autoencoder Experiment

As a side experiment, we have investigated how an autoencoder performs on the sensor time series used in the LSTM experiments. According to [21], autoencoder is the only identified deep learning based method for anomaly detection in videos and video can be seen as a multivariate time series with very many dimensions.

B.1 Background

An autoencoder [39] is a specific type of neural network where the goal is to teach the network the key features of some data and represent it in a less dimensional space, and then reproduce the original data from that representation. These tasks are handled by the encoder and decoder parts respectively. Figure B.1 shows the architecture of an autoencoder where in the encoder each layer has fewer neurons than the last and vice versa in the decoder part. The input and the output must have the same dimensions. A commonly used variation of the autoencoder is the denoising autoencoder, used in [44, 28]. Instead of using normal data input, noise is added to the input data while still using the clean version of the data for training. The idea is that this should force the autoencoder to learn more appropriate features.

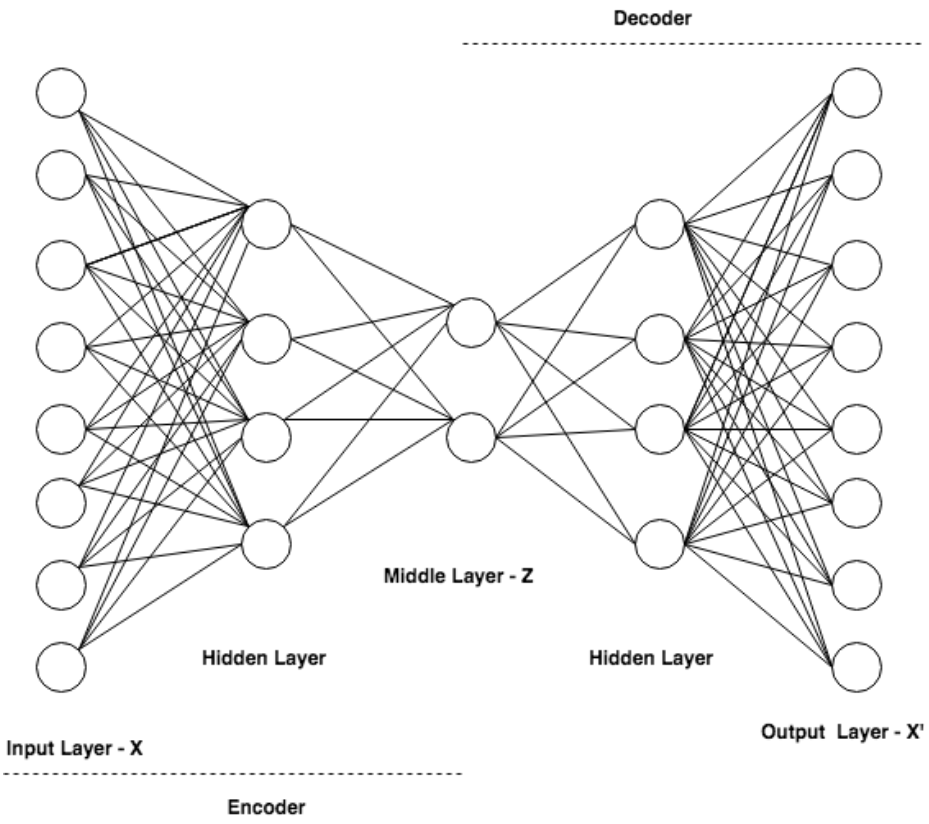


Figure B.1: Autoencoder architecture

B.2 Related Work

In this section we present the related work we found where autoencoders were used. One of them is used for anomaly detecting in multivariate time series much like we are doing, while other one is used for anomaly detection in video. Because video can be seen as a very high dimensional time series, it is still relevant for our work.

B.2.1 Anomaly Detection using Autoencoders with Nonlinear Dimensionality Reduction

In the article from [40] both an denoising autoencoder and a normal autoencoder is used to identify anomalies in addition to two versions of PCA. They tested their approaches on

both real data from satellites and artificial data created from a Lorenz system simulation. They do not specify the architecture of their autoencoder, but they say that their datasets from satellite A and satellite B contains 17 and 106 sensors respectively. It is reasonable to assume this denotes the number of neurons in both their input and output layers. They do not specify the mean values of the recreation error over the normal data, but from the graphs it seems to be somewhere between 0.5 and 1.0 for the normal autoencoder on satellite A and around 0.5 on satellite B. We will later use these values to compare to our own results. On data from satelite A, the normal autoencoder performed as bad as linear PCA on the normal data, but the error were still higher on the anomalous data. The denoising autoencoder on the other hand, had much lower error on the normal data while having the same high error on the anomalous data. Kernal PCA had somewhat higher error on normal data, but the lowest of all on the anomalous data. On satellite B, all approaches had similar error when comparing the recreation of normal data against anomalous data that they were indistinguishable from each other. The normal autoencoder and linear PCA had a small elevation on the anomalous data.

B.2.2 Learning Deep Representations of Appearance and Motion for Anomalous Event Detection

In the the work from [46], they use use fully unsupervised techniques to identify anomalies in video surveillance. They use two separate denoising autoencoders for encoding raw image pixels and motion representation. The outputs of these are merged into one representation. Like in our windowed approach they also reduce the number of neurons by half until they reached the bottleneck layer with the fewest neurons. Both in the image and motion representation they use sliding windows to include contextual data over a time span. After this they use a support vector machine (SVM) to classify patches of the images into categories which are combined to give the final score. Given the methods they picked out to compare their results to, they seem to get promising results. Even though the autoencoder parts of their architecture are comparable to our, the metrics they provide can not in any way be compared to our results.

B.3 Experiments

The following experiments have been conducted with the use of an autoencoder:

- Regeneration whole state using all sensors at one timestamp
- Window regeneration using a sequence of 100 for a single sensor

B.3.1 Predicting the Full State of Operation

Since we used a collection of sensors we know are in proximity, all the values from sensors at a certain time can be thought of as the current state of operation in that system. We make the assumption that most of the operation in our dataset is what can be considered normal operation. If that is true, a regeneration which deviates from the input data will happen when the model receives a state that deviates from most of the data it has been trained on.

This technique regenerates values from every input of the network, meaning that the error of regeneration does not apply for only one sensor, but rather all sensors that are fed into the model. If we use a LSTM network we can choose the sensor it is predicting, but for the autoencoder it will be punished equally for prediction faults in all output sensors. At the same time we can see the error for all sensors from only one trained model, and might be able to tell which of all the 24 sensors that in behaving anomalous.

We think it very important that noise is not added to the input of this model because then it might be able to recreate a normal state from input where one sensor show anomalous values. Especially things like adding salt and pepper noise, which is done by setting random values to 0, could have the exactly opposite effects of what we want.

B.3.2 Windowed Single Sensor

Using a sequence of values from one sensor and running it through an autoencoder requires no knowledge of the domain. It may only tell us whether some time interval of operation from one specific sensor is something similar to what is seen in the training data. Sliding window with a size of 100 values is the input data for this autoencoder.

B.3.3 Autoencoder Training Configuration

Tables B.1 and B.2 shows the type of layers, number of units in each layer, and the activation function of the full state experiment and the window experiment. The training

hyperparameters for the autoencoder is shown in Table B.3.

Table B.1: Autoenoder: Full state

Layer	Units	Activation
Dense	24	ReLU
Dense	15	ReLU
Dense	5	ReLU
Dense	15	ReLU
Dense	24	Sigmoid

Table B.2: Autoencoder: Window

Layer	Units	Activation
Dense	100	ReLU
Dense	50	ReLU
Dense	15	ReLU
Dense	50	ReLU
Dense	100	Sigmoid

Table B.3: Training hyperparameters

Parameter	Value
Epochs	100
Optimizer	SGD
Loss	MSE

B.4 Results

The results for the autoencoder is presented in Table B.4.

The MAE and MSE are measured by using built in functions in Keras, and unlike the LSTM network they are measured on the 10,000 values after the training set ends. Concretely the MAE means that on average there is a 2.5% deviation between recreated data and real data. As mentioned before our hope is that when trying to recreate data from anomalous parts of the dataset this will be harder to recreate and thus give higher error. The MAE and MSE are measured over all sensors so this means that one sensor can have a very high error while the others have smaller error.

Table B.4: Normalized MAE and MSE from the next 10000 steps after the training set

Model	MSE	MAE
State regeneration	0.0013	0.0254
Windows (Temperature)	0.0010	0.0295
Windows (Pressure)	0.0010	0.0297

B.5 Discussion

B.5.1 Full State Generation

For the full state generation autoencoder to be useful we need it to indicate which sensors are behaving most unusual at any given time. We think that since the whole state is compressed to be represented by 5 neurons in the middle layer, it might lose the ability to detect if a single sensor among the 24 sensors is behaving abnormal. This could manifest itself as a disturbance among recreation of other sensors. Our hypothesis was that if a small part of the state was too different from the training data, we would see large error in multiple sensors.

In Figure B.2 all the sensors except the faulty ESV is plotted. A plot of full state generation including the ESV sensors can be seen in Appendix C. The plot shows the **difference** of the recreated sensor values and the real input values on the whole set of unseen data. There are some spikes in the beginning, and after these it seems like almost all sensors start to behave a bit more suspicious. After the known anomaly it seems like some sensors are going back to normal, while at least three have elevated levels of recreation error. One can also see that a long time before the incident, after the third spike, all sensors have somewhat increased error which could indicate that there is something wrong with the system.

According to this plot the whole last part of the operation is in a relatively unknown state. This includes our whole test set used in the results for the LSTM section. There seems to be even higher error than during the anomaly, even though this section is considered normal operation of the system.

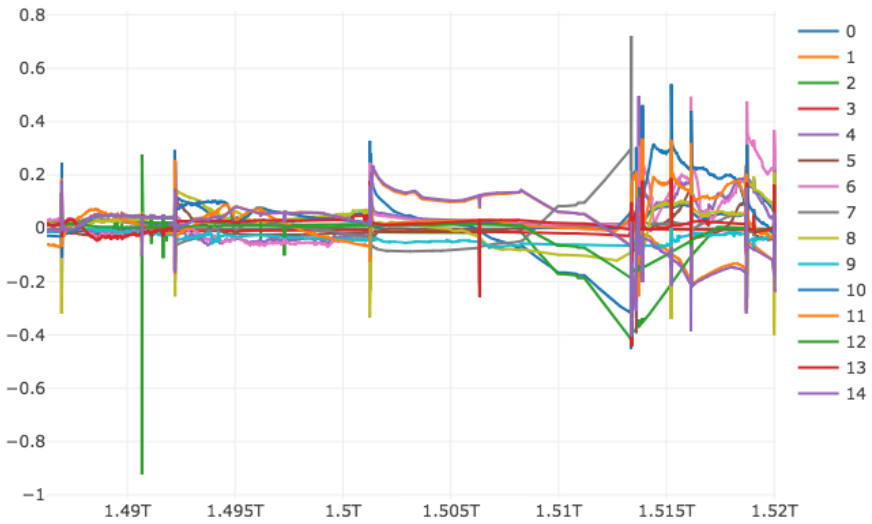


Figure B.2: Regeneration of state

Regeneration of the Pressure and Temperature Sensor

Figures B.3 and B.4 show the plots of the reconstruction error and the reconstructed values of the pressure sensor and temperature sensor that is predicted. When looking at the green graph in both figures, the models seem to have large reconstruction errors every time the actual value spikes significantly. Another thing to notice is how the reconstruction values are lower than the actual values for a period after the spikes that are prior to the known anomaly. For the spikes after the anomaly a decreasing trend prior to the spikes can be seen in B.4, while the reconstruction is more unstable for the whole period in B.3. Since each output is only based on the input from one time step, the model is not affected by the spike of values that has occurred before. This suggests that there is some other subset of sensors which are affected by the spike, since the temperature sensor goes straight back to normal operation after it.

What Happens after the Drop?

We wanted to find out why the reconstruction of the pressure and temperature sensor is slowly increasing rather than immediately going back to normal operation, like the actual

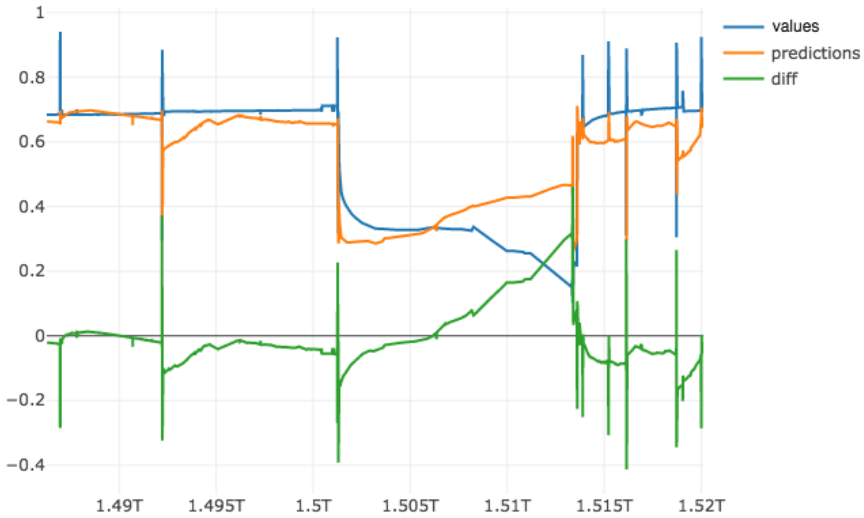


Figure B.3: All sensors autoencoder plot - pressure

values after the spike occurring approximately at 1.4925T. A possible reason for this might be that there are other sensors that are causing this. We plotted a zoomed in version of the spike in Figure B.4 which can be seen in Figure B.5. In addition, plots of all other sensors separated into to plots of for the PI/PT and TI/TT sensors in Figure B.6, and the ESV sensors in Figure B.7. All values were normalized to be between zero and one so they can be plotted in the same figure.

When looking at the changes in temperature and pressure in Figure B.6 it is hard to find any significant difference that could cause the effect we are seeing in the plot in Figure B.5. It might be that the values do change after the incident but that they are hidden in the normalization. Some of the sensors have a few data points that are either much bigger or smaller than the rest which causes the rest of the plot to be squashed into a small range in the plot. These values could be because of malfunction in the sensors, but we do not know that for sure. The model can still be very sensitive to small changes in these sensors because of the bias of the neuron can adjust the threshold from where the neuron starts to activate. At the same time the neurons in the autoencoder do not work like the ones in the LSTM so they cannot be explained by gates that are opened during the drop because as mentioned, there is no notion of time is this model.

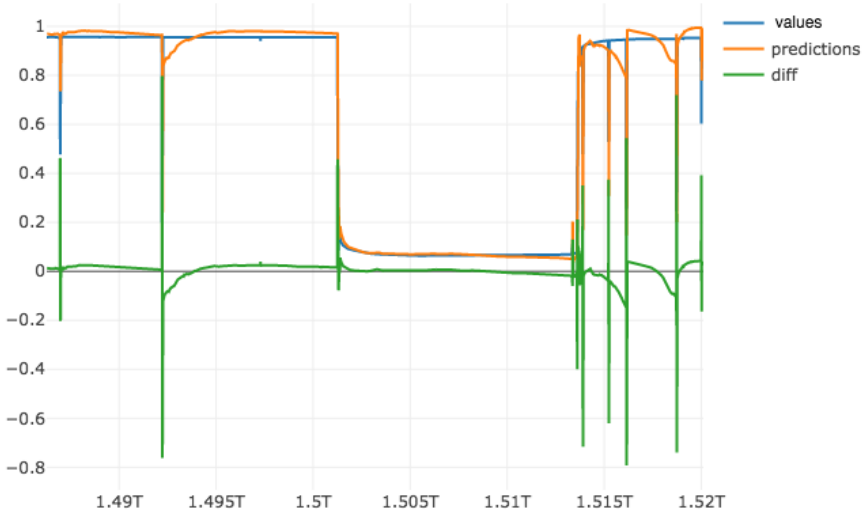


Figure B.4: All sensors autoencoder plot - temperature

So we see that there is no sensor which seems to be causing the drop alone. Some of them spend some time before getting back to normal values after the drop, specifically two ESV sensors, but they also have linearly decreasing values prior to the spike which is not visible in the regeneration. This may indicate that they do not impact the model when their values are decreasing, but do impact it when increasing, or are not impacting the model at all. In other words, the values of the sensors do not give a clear indication for this observation, something that we hoped for them to do.

B.5.2 Windowed Generation

Using an autoencoder to regenerate windows of values from single sensors did not yield the results we hoped for. All the generated data had a relatively large constant offset from the actual values. Like when predicting when using faulty input in the LSTM, the plotted results from this experiment showed a very clear repeated pattern throughout the whole set of unseen data. The almost exactly same pattern was repeated every 100th timestep, only changing starting point and amplitude according to previous values. In Figure B.8 we can the see plot of the original data, the regenerated data and the difference from the

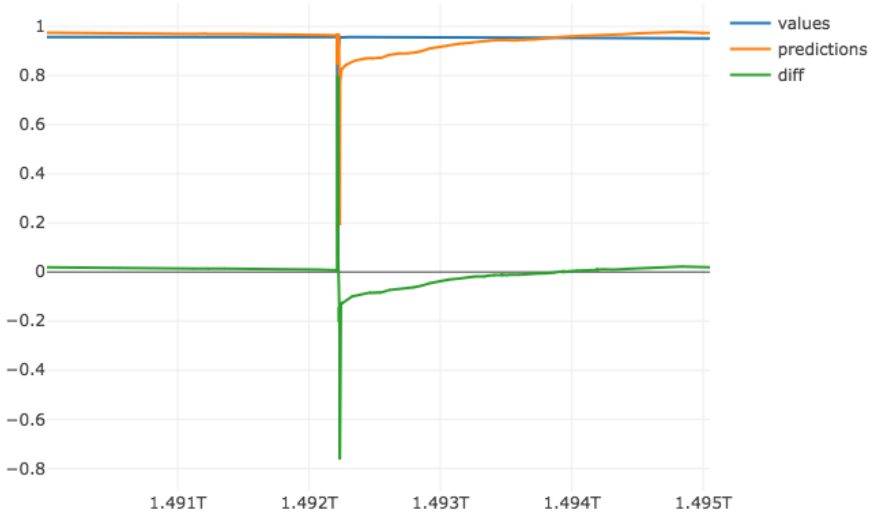


Figure B.5: Prediction of temperature sensor

windowed autoencoder on the unseen data. Like in the other plots that are forecasting six time steps, we only used every 100th batch of values and merged them together. While the error peaks higher during the data for the known anomaly, we still wanted to see some more nuances in the regenerated data elsewhere.

B.5.3 Comparison to Related Work

When trying to compare this to the results from the related work where they used autoencoders on satellite sensors [40] we found a number of challenges. One of them is that their plots do not use normalized data while our measures do. They do not provide the average results over their test and anomaly sets and they do not clearly show their max values so it is not possible to do a rough calculation of the percentage of their error. There are also some differences between the neural networks used in our experiments and in their. They use sigmoid activation functions in their hidden layers, while we use ReLU. Also they use a linear activation in the output layer because they do scale input which implies the output label is not scaled, unlike in our experiments where input is normalized between 0 and 1. They use an linear activation in the output layers since their output values are not scaled,

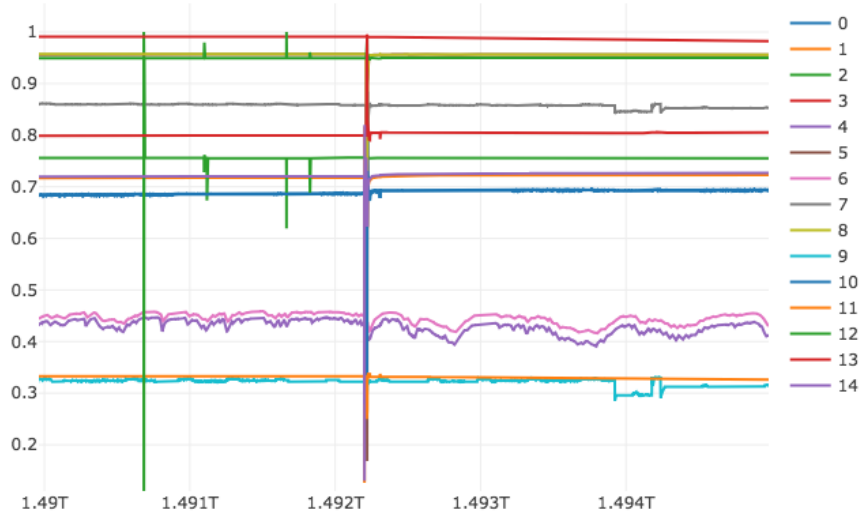


Figure B.6: Temperature and pressure sensors

while we use sigmoid.

In the experiments with satellite data they use denoising autoenders by using salt and pepper noise on the input. This means that they some amount of the input to zero without changing the output values. This makes the model learn to recreate the denoised values from noisy values. In our case we think this could have had opposite effect. We want to be able to know when one or more sensors is behaving abnormal, but a denoising is trained to ignore it. If anything we think that training the autoencoder to be sensitive for any abnormalities could give better indication of anomalies. This would probably requires a much cleaner dataset and making sure that all the input sensors are correct.

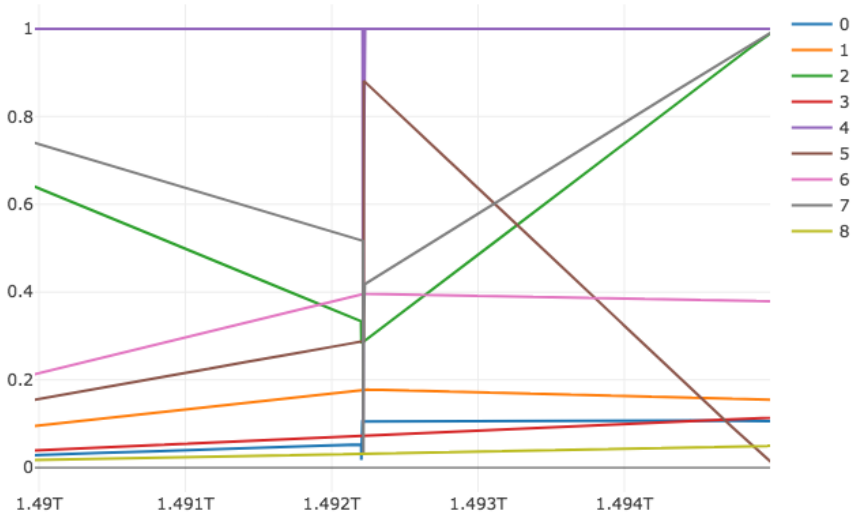


Figure B.7: ESV sensors

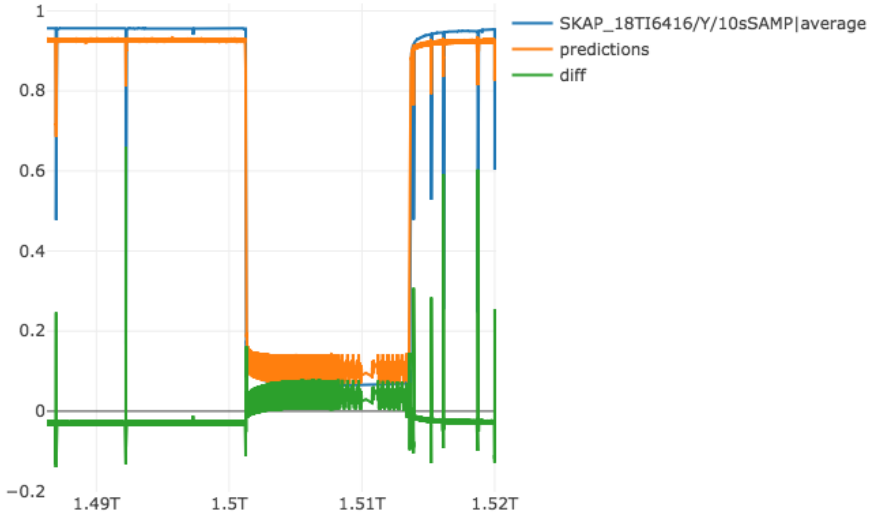


Figure B.8: One windowed sensor autoencoder plot

Appendix C

Extra plots

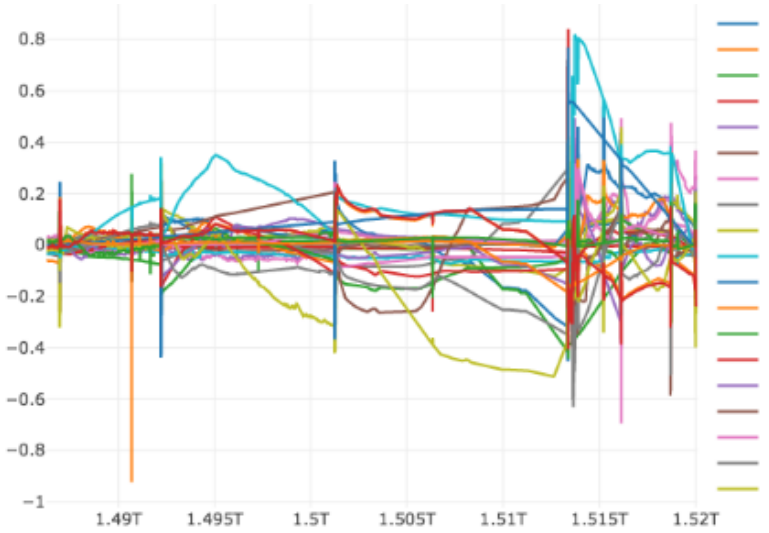


Figure C.1: Full state generation including ESV sensors

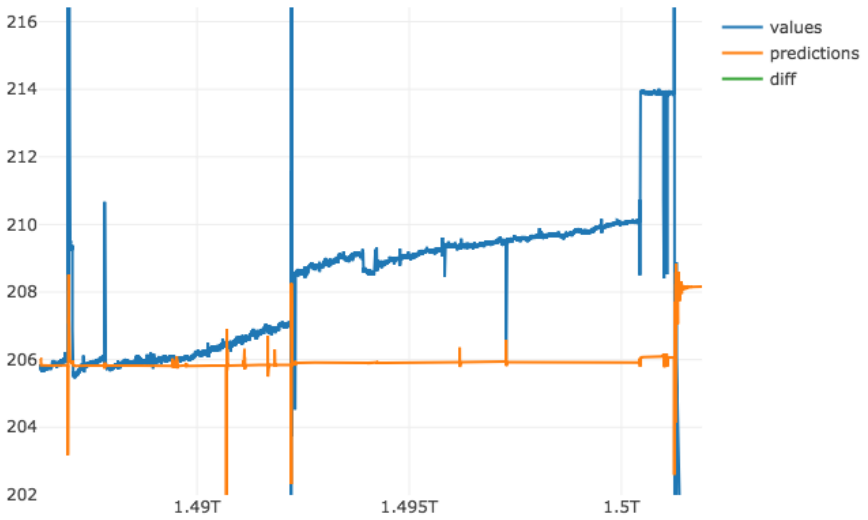


Figure C.2: First four sensors - pressure - one time step

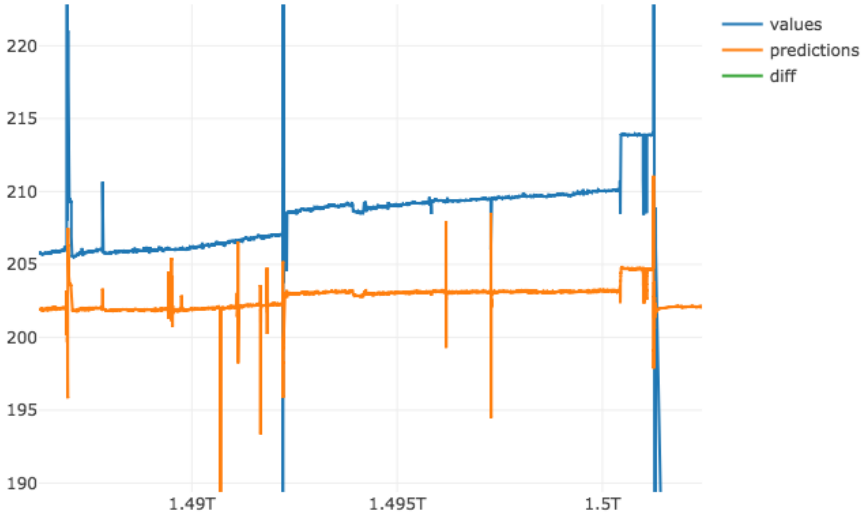


Figure C.3: First four sensors - pressure - six time steps

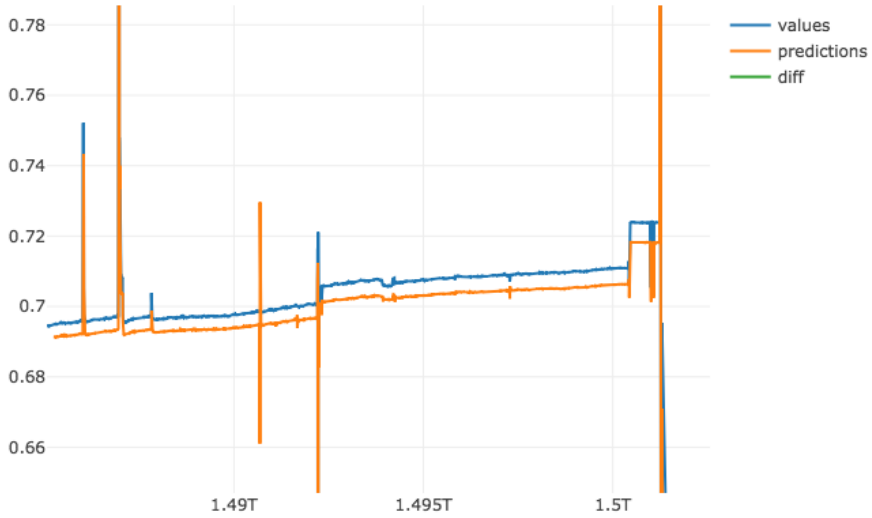


Figure C.4: PT/PI + ZI - pressure - 1 hour - one time step

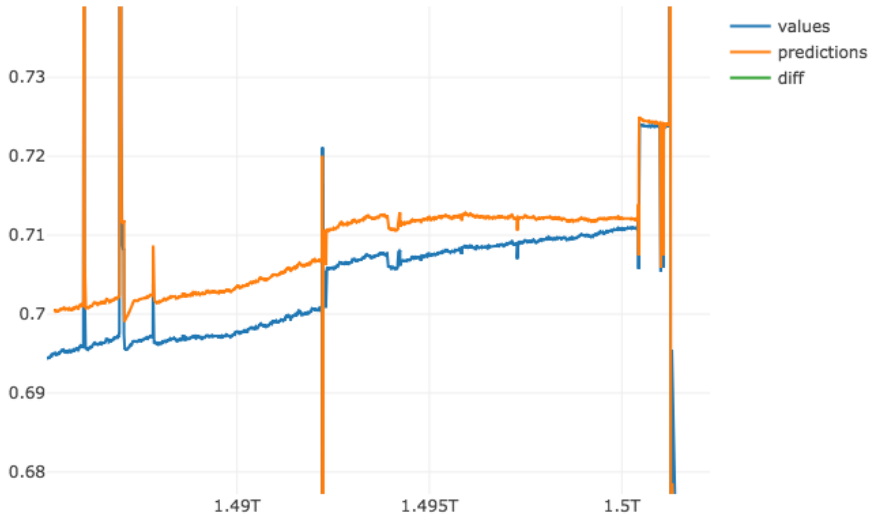


Figure C.5: ESV + ZI - pressure - 1 hour - one time step

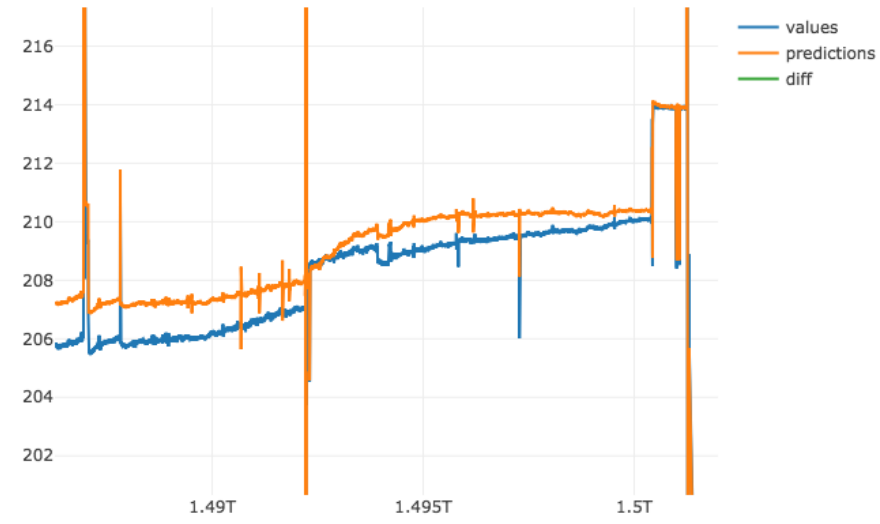


Figure C.6: TT/TI + ZI - pressure - 10 minutes - one time step

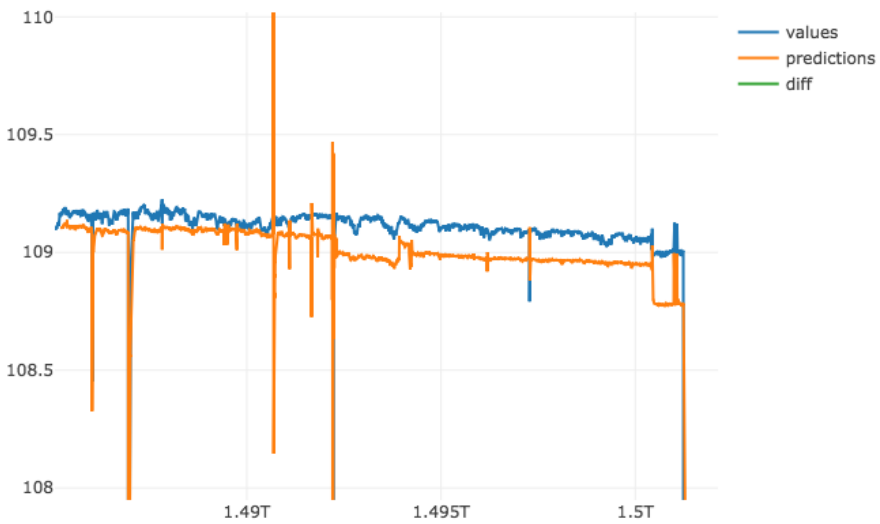


Figure C.7: PT/PI - temperature - 1 hour - one time step

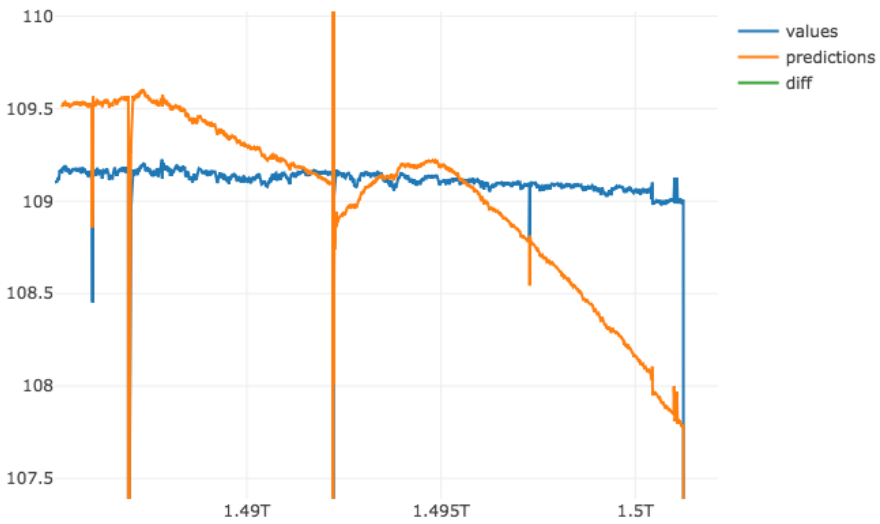


Figure C.8: ESV - temperature - 1 hour - one time step

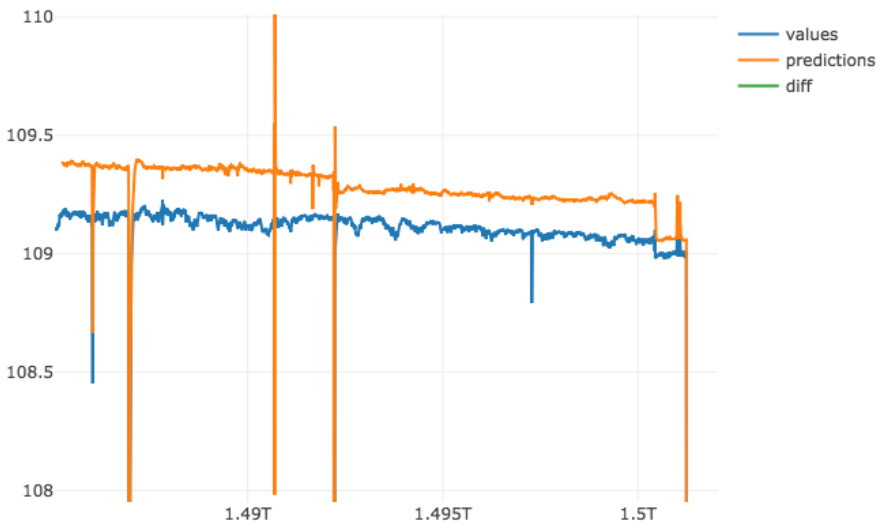


Figure C.9: PT/PI + TT/TI - temperature - 1 hour - one time step

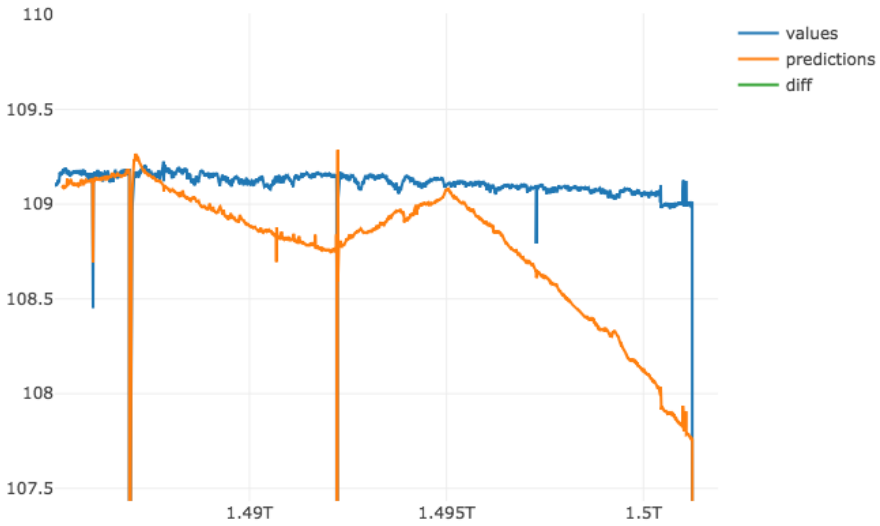


Figure C.10: TT/TI + ESV - temperature - 1 hour - one time step

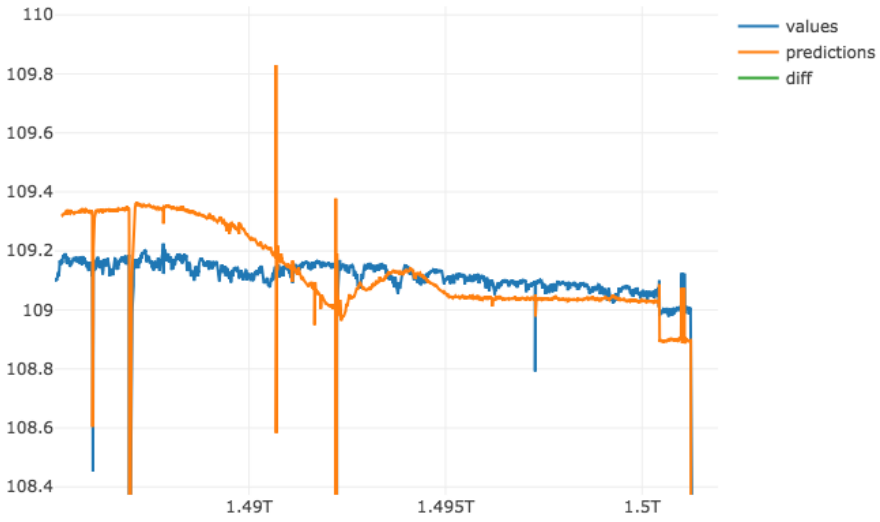


Figure C.11: Prior sensors - pressure - 1 hour - one time step