Abstract

Heat exchangers are among the key components in oil and gas processing, by facilitating heat transfer between separated fluids. Optimal utilisation of this capability is necessary to ensure high energy efficiency in processing facilities. During operation, heat exchangers experience accumulation of unwanted material on the heat transfer surfaces, which reduces thermal conductivity and hinders fluid flow. Because of this, implementing condition monitoring to estimate heat exchanger performance is vital. Traditional monitoring techniques have proven unreliable in practice. Hence, Equinor is looking to incorporate machine learning methods in the evaluation of heat exchanger performance. Additionally, it is desirable to determine a minimal set of sensors that suffice to monitor heat exchangers for future, potentially unmanned, processing facilities.

This thesis proposes a set of predictive models, using a reduced number of input parameters to estimate heat exchanger performance by calculating the deviation between predicted and measured coolant outlet temperature. A large selection of linear, multilayer perceptron and recurrent neural network regression models are tested for three different processing facilities. This is supported by the implementation of a software module, enabling rapid prototyping of, and comparison between, machine learning models.

Several discoveries are made which highlight the difficulty of applying machine learning methods to heat exchanger monitoring, most importantly continuous decreases in system performance and lack of known performance measurements. Data obtained through process simulations are used to evaluate the capabilities of the predictive models for cases of known system degradation, for which encouraging results are obtained. When applied for real processing facilities, similar patterns are observed.

Overall, the proposed models are found to estimate heat exchanger fouling very well. Even so, much work remains before these findings can be used for maintenance planning in practice. Numerous topics for further development are identified, most importantly the inclusion of the proposed predictive models in a robust preventive maintenance scheme.

Sammendrag

Varmvekslere er blant de viktigste komponentene brukt under olje- og gassprosessering, ettersom de muliggjør utveksling av varme mellom ulike fluider. Optimal varmeoverførsel er viktig for å maksimere energieffektiviteten for prosesseringsanlegg. Under bruk oppstår det begroing på overflatene inne i varmeveksleren, som reduserer varmeoverføringsevnen og hindrer flyten av fluid. På grunn av dette er det ønskelig å monitorere ytelsten til varmevekslere kontinuerlig. Tradisjonelle monitoreringsteknikker har vist seg å være upålitelige i praksis. Derfor ønsker Equinor å implementere maskinlæringsmetoder for å måle ytelsen i varmevekslere. I tillegg er det ønskelig å identifisere et minimalt antall sensorer som bør installeres på nye anlegg for tilstrekkelig overvåkning.

Oppgaven foreslår fem ulike prediktive modeller. De benytter et ulikt antall inputparametre for å estimere ytelse ved å relatere begroing til avviket mellom predikert og målt utløpstemperatur på kjølesiden av varmeveksleren. Flere ulike regresjonsmodeller, blant annet lineære modeller, neurale nettverk og neurale nettverk med tilbakekoblinger, har blitt testet for tre ulike prosesseringsanlegg. For effektiv implementasjon og sammenlikning av ulike modeller har en omfattende programvarepakke blitt utviklet.

Flere interessante oppdagelser har blitt gjort som belyser ulike utfordringer ved å benytte maskinlæring for data fra varmevekslere. Dette gjelder særlig kontinuerlig reduksjon av ytelse, samt mangel på ytelsesmålinger. På bakgrunn av dette har de prediktive modellene blitt testet på data fremskaffet gjennom prosessimuleringer, hvor mengden begroing kan legges til eksplisitt. Simuleringene utgjør dermed en fasit, og ytelsen til modellene kan måles utfra hvor godt de etterlikner nivået av begroing. Resultater for disse forsøkene er svært lovende. Bruk av de prediktive modellene på data fra ekte prosesseringsanlegg viser liknende ytelsesmønster.

De foreslåtte metodene anses å være svært egnet for å estimere mengden begroing i varmevekslere. Likevel gjenstår det mye arbeid før dette kan benyttes til vedlikeholdsplanlegging i praksis. Avslutningsvis presentes en rekke forslag til videre arbeid, hvorav utvikling av en prediktiv vedlikeholdsmodell basert på de foreslåtte overvåkningsteknikkene er ansett som høyeste prioritet.

Preface

This thesis concludes my Master of Science degree at the Norwegian University of Science and Technology (NTNU), as part of the Engineering and ICT study program with a specialization in ICT and Mechanical Engineering. The presented study is a collaboration between Equinor ASA and the Department of Mechanical and Industrial Engineering (MTP), focusing on the application of machine learning methods for the oil and gas domain.

I would like to thank each of my supervisors at Equinor for their contributions during my thesis work. Even Solbraa, for helping me with numerous topics within oil and gas processing. Knut Maråk, for his continued aid in identifying and explaining challenges related to heat exchanger monitoring. Marlene Louise Lund, for providing me with valuable assistance while navigating the vast Equinor IT infrastructure. Also, many thanks to Bjørn Haugen for his contributions during my thesis work and throughout my time at MTP. Last but not least, I am grateful for the continued support of my parents, Dag and Eva.

Applying machine learning to the field of oil and gas performance monitoring has proven both challenging and rewarding. I hope that the findings presented in this thesis will facilitate and inspire further work on predictive maintenance using machine learning within Equinor and elsewhere.

I will reminisce my time at NTNU with great fondness, and am grateful for the many friends I have made along the way.

Trondheim, June 3rd 2020

Herman Wika Horn

Contents

Figures

Tables

Code Listings

Abbreviations

Index	Description
AdaBoost	Adaptive Boosting
AdaGrad	Adaptive Gradient
Adam	Adaptive Moment Estimation
AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
CBM	Condition Based Monitoring
CPU	Central Processing Unit
CSV	Comma Separated Values
CUDA	Compute Unified Device Architecture
CV	Coefficient of Variation
DBN	Deep Belief Network
EoS	Equation of State
GPM	General Path Model
GRU	Gated Recurrent Unit
GPU	Graphics Processing Unit
KPI	Key Performance Indicator
LASSO	Least Absolute Shrinkage and Selection Operator
LMTD	Log Mean Temperature Difference
LNG	Liquid Natural Gas
LPG	Liquid Petroleum Gases
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MLP	Multilayer Perceptron
MSE	Mean Squared Error
NGL	Natural Gas Liquids
NTU	Number of Transfer Units
PCA	Principal Component Analysis

Index	**Description**
PCHE	Printed Circuit Heat Exchanger
PHE	Plate Heat Exchanger
PR	Peng-Robinson
PT	Pressure Temperature
PV	Pressure Volume
ReLU	Rectified Linear Unit
RMSE	Root Mean Squared Error
RMSProp	Root Mean Square Propagation
RNN	Recurrent Neural Network
RSS	Residual Sum of Squares
RUL	Remaining Useful Life
SD	Standard Deviation
SGD	Stochastic Gradient Descent
SRK	Soave-Redlich-Kwong
STHE	Shell-and-Tube Heat Exchanger
SVM	Support Vector Machine
SVR	Support Vector Regression
VPN	Virtual Private Network

Nomenclature

Symbol	Description	Unit
A	Area	m^2
c_p	Specific heat capacity at constant pressure	J/kgK
C	Heat capacity rate	W/K
C_f	Mass Flow Correlation Factor	$-$
C_{max}	Maximum of C_c and C_h	W/K
C_{max}	Minimum of C_c and C_h	W/K
C_r	Heat Capacity Ratio	$-$
D	Hydraulic diameter	m
E	Heat exchanger effectiveness, error function	$-$
$E[x]$	Expectation of variable x	$-$
f_d	Darcy friction factor	$-$
h	Convection heat transfer coefficient	W/m^2K
k_b	Boltzmann constant	J/K
k	Thermal conductivity	W/mK
L	Length	m
$LMTD$	Log Mean Temperature Difference	$K, ^\circ C$
$\dot{m}$	Mass flow	kg/s
n	Number of moles	$-$
N_a	Avogadro constant	mol^{-1}
N	Number of samples	$-$
p, P	Pressure	Pa
q	Heat	W
$q_x^{''}$	Heat flux	W/m^2
Q	Volume flow rate	m^3/s
R	Thermal resistance	m^2K/W
R^2	Coefficient of determination	$-$
R_g	Universial gas constant	$J/K \cdot mol$
$Tanh$	Hyperbolic tangent function	$-$

Symbol	Description	Unit
T	Temperature	$K, °C$
T_s	Surface temperature	K
T_∞	Fluid bulk temperature	K
T_{sur}	Surrounding temperature	K
U	Overall heat transfer coefficient	W/m^2K
v	Mean flow velocity	m/s
$v_{\%}$	Valve opening	$-$
V	Volume	m^3
V_m	Molar volume	m^3/mol
$\boldsymbol{w}$	Weight vector	$-$
$\boldsymbol{x}$	Input vector	$-$
$\boldsymbol{y}$	Output vector	$-$
$\Delta p, dP$	Pressure difference	Pa
$\Delta T, dT$	Temperature gradient	K
$\Delta x, dx$	Length of transfer	m
∇	Gradient	$-$

Greek letters	Description	Unit
δ	Propagation error	$-$
ϵ	Emissivity, error	$-$
ϵ_{rad}	Radiative efficiency	$-$
η	Learning rate	$-$
μ	Mean	$-$
$\mathcal{N}$	Normal distribution	$-$
ρ, σ	Activation function, standard deviation	$-$
σ_b	Stefan Boltzmann constant	W/m^2K^4
ϕ	Basis function	$-$

Subscripts	Description
c	Coolant, cold
d	Design
f	Fouling
h	Hot
i	Inlet, index i
j	index j
o	Outlet
p	Process
tot	Total

Chapter 1

Introduction

This chapter introduces the thesis objectives and relevant background regarding central topics. The approach chosen to fulfil these objectives is presented, along with the limitations that apply. Lastly, an outline of the thesis contents is given.

1.1 Background

The petroleum industry operates many complex, large-scale industrial processes and technologies. Producing hydrocarbon products with widespread use in an optimized and safe manner requires thoughtful treatment in diverse stages. Various types of equipment are used to transport, separate, heat, cool, and otherwise handle the hydrocarbon fluids. Among others, heat exchangers are vital in ensuring high energy efficiency in processing facilities, by facilitating heat transfer between fluids. Usually, a hydrocarbon process fluid is cooled by the use of some coolant. This is done so that specified temperatures and pressures required at various stages of production may be obtained.

During operation, heat exchangers experience accumulation of unwanted material on the heat transfer surfaces. This is referred to as fouling, and reduces the thermal conductivity of the heat exchanger, while also hindering fluid flow and causing increased pressure drop. When thermal conductivity drops, the coolant demand increases. Eventually, the coolant demand may exceed the possible coolant supply, at which point the heat exchanger can no longer maintain the process outlet temperature required by downstream equipment. Alternatively, pressure drop may exceed the maximum thresholds of the heat exchanger. In either case, maintenance in the form of heat exchanger cleaning must be performed.

Traditional monitoring techniques for heat exchangers rely on the estimation of heat exchanger effectiveness or overall heat transfer coefficient. This is obtained through thermodynamic calculations. For such calculations to be carried out in practice, several assumptions and estimations about the fluid parameters must be

made. This poses multiple challenges, limiting the performance of the methods. Lack of measuring equipment for the coolant fluid is a recurrent problem, especially for old facilities. Installing a minimal number of sensors is often preferred for processing facilities in general, in an effort to reduce costs and break-even prices. Due to the shortcomings of traditional techniques, the implementation of more reliable performance estimators is desired.

Equinor is looking to build unmanned production platforms, for which predictive maintenance using remote monitoring techniques is critical. Faults must be anticipated ahead of time, so that inspections and repairs can be performed preventively with limited economical consequences and risk, as opposed to in a corrective manner after failure has already occurred. In relation to this, a research project focusing on condition monitoring of heat exchangers was started in 2019. A master thesis for this project by Jensen and Nordhus [1] found limited success monitoring heat exchangers using traditional techniques. Recommendations were made that machine learning methods may be applicable. The research project is continued through the writing of this thesis.

1.2 Problem definition and objectives

The problem statement for the thesis work is as follows:

> Heat exchanger fouling has proven difficult to estimate using traditional monitoring techniques. Fouling estimation is vital in ensuring maintenance can be performed in a preventive rather than corrective manner. For new facilities in particular, it is desirable to implement condition monitoring of heat exchangers using a limited set of sensors. Based on the success of machine learning in order fields, its capabilities for the oil and gas domain should be investigated. More specifically, condition monitoring of heat exchangers should be implemented using appropriate machine learning methods, to enable the use of preventive maintenance schemes.

Based on the problem statement, the following thesis objectives are defined. These are the objectives which the thesis results, discussion and conclusion will aim to achieve and provide sufficient answers for.

1. Determine the applicability, and potential limitations, of machine learning methods for the oil and gas domain.
2. Develop and implement machine learning methods capable of estimating the fouling factor in heat exchangers, to assist with preventive maintenance decision making.
3. Determine necessary measuring equipment to accurately estimate heat exchanger fouling for future, potentially unmanned, processing facilities.
4. Facilitate further research within these topics in Equinor.

1.3 Approach

Literature and related work is reviewed extensively to identify widespread best practice monitoring and modelling techniques. Relevant machine learning methods for the heat exchanger domain are evaluated, before predictive models to estimate heat exchanger performance are derived. Model applicability is justified through the use of heat exchanger theory. Two hypotheses are formulated, suggesting that heat exchanger monitoring may be possible using machine learning regression models and a reduced number of sensors.

Data for processing facilities with different operating conditions are acquired and analyzed. The predictive models are applied in order to investigate their ability to estimate heat exchanger fouling. To facilitate the implementation and comparison of machine learning models, an extensive programming framework is developed. Results are evaluated based on empirical interpretation and calculated metrics.

1.4 Limitations

To achieve the thesis objectives, interdisciplinary work spanning process engineering, machine learning, and maintenance theory is required. It is infeasible to cover all relevant topics. Hence, study of reliability and maintenance planning is neglected. This is reflected in the thesis results and conclusions, by proposing predictive models and a foundation for future work rather than a complete maintenance solution. For the oil and gas industry, and heat exchangers in particular, limited research concerning the use of cutting edge machine learning is found. Available work appears to use small datasets and simple algorithms, indicating that deep learning for heat exchanger monitoring is a relatively untouched field.

Machine learning in itself is a relatively new field with considerable ongoing research, for which improved algorithms are proposed at a rapid pace. Large increases in model complexity are often introduced to improve performance by a small margin. Optimization of machine learning algorithms for many practical purposes is not yet well understood. As a result, this thesis implements what is found through related work and theoretical review to be the widespread best practice, without exploring many of the advanced techniques proposed in literature.

Model training is performed using limited hardware, resulting in lengthy running times for algorithms and limitations on applicable architectures. Note that machine learning experiments found in research papers and other sources are usually trained on systems with far greater computational power.

Model performance is evaluated based on knowledge obtained throughout the thesis work, and with the help of Equinor research personnel. Although supervising facility engineers were presented with the thesis results and have expressed optimism, the true quality of the models can only be evaluated when applied for new facilities and used to estimate fouling in practice.

1.5 Outline

Theory and related work

Chapter 2 gives a thorough review of related work, specifically for previous research in Equinor, machine learning, predictive maintenance, and heat exchanger fouling. These topics are covered in sections 2.1, 2.2, 2.3 and 2.4, respectively.

Chapter 3 explains necessary theory regarding oil and gas processing, heat exchangers, and machine learning, in sections 3.1, 3.2 and 3.3, respectively. Each topic is covered extensively, so that readers with expertise within either domain may grasp the thesis as a whole.

Methodology and implementation

Chapter 4 presents the proposed methodology and predictive models for heat exchanger monitoring, as well as the applied methods and evaluation metrics.

Chapter 5 describes the implemented software module. This chapter is quite brief compared to the extent of the implementation, as the implementation itself is not vital in presenting or understanding the thesis objectives and results.

Discoveries and results

Chapter 6 highlights notable discoveries made while extracting, analyzing and treating the acquired data.

Chapter 7 presents the results, obtained by applying the predictive models to heat exchanger facilities. A large number of model predictions are plotted, and ability to detect fouling is discussed.

Discussion and conclusions

Chapter 8 discusses the thesis results, highlighting particular challenges and important considerations when evaluating the applicability of the proposed methodology.

Chapter 9 concludes the thesis, and lists recommendations for future work.

Appendices

Appendix A contains additional theory. **Appendix B** presents code for extraction and preprocessing of data. **Appendix C** explains the process of selecting applicable neural network hyperparameters, as well as a technique for estimating model uncertainty for neural networks. **Appendix D** shows additional results not included in Chapter 7. **Appendix E** lists statistical data for the facilities used throughout the thesis.

Chapter 2

Related work

This chapter presents an extensive review of academic papers discussing machine learning, predictive maintenance, and heat exchanger fouling. These topics are covered in sections 2.2, 2.3 and 2.4, respectively. For each paper, details regarding approaches, results and important remarks are highlighted. Relevant topics such as machine learning methods, implementation and use of deep architectures, regularization in neural networks, anomaly detection, heat exchanger faults, and tuning of machine learning model parameters are discussed.

Before that, a brief summary of a master thesis preceding this thesis as part of the same Equinor research project is presented in section 2.1. With a focus on fault detection and fault prevention in heat exchangers, Jensen and Nordhus [1] attempt to identify heat exchanger faults through the use of thermodynamic modelling. Some machine learning methods are also applied in order to investigate the potential of machine learning for heat exchanger systems.

Finally, a brief summary of the preceding sections is presented in section 2.5. Note that much of the theory presented in this chapter is not explained in detail immediately, but rather greatly expanded on in Chapter 3.

2.1 Research at Equinor

Master thesis 2019

As a predecessor to this project, Jensen and Nordhus [1] explore six approaches to condition monitoring of heat exchangers. It highlights that heat exchangers currently in use within the industry lack standardized condition monitoring methods. This leads to the use of statistical or experience-based maintenance in practice, rather than more modern maintenance approaches. Lack of or inaccuracies in measurements within the data foundation used throughout the thesis meant complications were encountered in acquiring satisfying results.

A detailed explanation of heat transfer fundamentals and fouling is given, describing the various forms of heat transfer, including conductive, convective and radioactive. Models for calculating heat transfer coefficients through fouled planes are presented, as well as the various fouling mechanisms. Several heat exchanger designs are discussed, before methods for calculating heat exchanger performance are presented.

In studying the current heat exchanger monitoring methodologies used in the industry, the thesis finds that typical approaches involve calculation of the overall heat transfer coefficient and thermal efficiency. With regards to machine learning, principal component analysis is highlighted as a mean of detecting and isolating abnormal conditions. Additionally, regression analysis and neural networks are discussed and said to have shown encouraging results in previous research. Alternative approaches are also explored, without significant findings.

The following methods of condition monitoring are selected for further research:

1. **Data Dashboard**: Illustrative solution to display and visualize commonly used performance variables, which must then be analyzed by operators.
2. **Duty per Valve Opening**: Performance indicator using heat exchanger duty divided by coolant valve opening. Assumed to be easy to calculate due to low number of unknowns.
3. **Thermal Efficiency**: Using the Effectiveness-NTU method, in which the ratio between overall thermal conductance and smallest heat capacity rate is calculated. Additionally, the Log Mean Temperature Difference (LMTD) method is explored.
4. **Principal Component Analysis**: MATLAB implementation of dimensionality reduction through Principal Component Analysis (PCA).
5. **Artificial Neural Networks (ANN)**: MATLAB implementation predicting desired output features based on chosen input features. Primarily tested with gas flow rate as output.
6. **Classification Learners**: MATLAB toolbox with various classifier algorithms.

1. Data Dashboard: The thesis concludes that the considerations made by the operator in the case of an operator-interpreted data dashboard may be too complicated, and therefore undesirable. Adding automotive capabilities to the dashboard would require manual analysis and definition of unwanted operation for each dashboard solution, making this method unreliable and inefficient.

2. Duty per Valve Opening: Considered easiest to implement, and therefore desirable if found to have acceptable performance. Gas flow rates and enthalpy calculations are made using HYSYS as simulation tool, before the heat exchanger duty is calculated. Valve opening of the cooling medium is measured directly. It is found that in order to interpret the Duty per Valve Opening measurement and derive meaningful information, several additional calculations have to be performed, increasing the complexity. No clear conclusions are made.

3. Thermal Efficiency, Number of Transfer Units (NTU): Once more considered easy to implement, requiring only temperature measurements and at least one flow rate as input parameters. Model uncertainty is introduced through estimation of thermal efficiencies and other system parameters, making accurate predictions challenging. Thermal efficiency for clean and fouled exchangers are compared, with inconsistent results, indicating that such methods are suboptimal. The conclusion is that the use of system design properties and estimation of thermal efficiencies make the method unsuitable for prediction of fouling.

4. Principal Component Analysis: Through analysis of the variance covered by each principal component, it is found that a significant portion of the variance, as much as 98 percent, can be explained using two principal components. By plotting the area of operation for these principal components in two dimensions, the hope is to distinguish regions of normal operation from instances of significant fouling. It is found that even during periods of fouling, the principal components lie within the expected value range as if there was no fouling. The majority of points with large deviation from the mean operating point are determined to originate from stops in production. Despite these negative results, further research is encouraged.

5. Artificial Neural Networks: A neural network implementation available in MATLAB is used to predict the process flow rate. This sensor was chosen due to a flow rate sensor malfunction, for which it was desirable to perform predictions. Results are encouraging, with somewhat accurate predictions of approximately 5-10 percent deviation from the measured values. However, areas of large deviation are also found. Neural networks are found to estimate flow rate with higher accuracy than Bayesian regression. This approach is considered the most promising of all the six methods, although it does not attempt to estimate fouling.

6. Classification Learners: Classification Learners in MATLAB are used in order to predict expected time until necessary maintenance. It is noted that such methods are highly dependent on accurate maintenance dates in the training set, meaning that if unnecessary maintenance is performed while the training set is recorded, this will be reflected by inaccurate predictions when using the trained model. The best results are obtained using a weighted nearest neighbour algorithm, although even this method shows significant variation and prediction inaccuracies despite capturing the most important maintenance trends. The results are considered promising.

To summarize, the thesis finds that the most important step, regardless of monitoring scheme, is to ensure sufficient instrumentation and sensor monitoring. Without adequate data, any monitoring scheme will have to rely on assumptions to some degree. Supervised machine learning is highlighted as the most promising method, although it is important to note that no steps are made towards fouling indication using these methods. It is also suggested that combining several methods, for instance the use of thermodynamic factors in machine learning methods, may prove useful.

2.2 Machine learning and statistical analysis

Hundman *et al.* [2] uses long short-term memory (LSTM) models and an anomaly detection system for multivariate time series data prediction. LSTM is highlighted as capable of maintaining memory of long-term dependencies, while the anomaly detection systems improves on the traditional detection schemes of predefined limits and manual analysis. A neural network of two hidden layers with 80 units in each layer, the Adam optimizer and dropout regularization is used.

Guyon and Elisseeff [3] gives an introduction to the benefits of feature selection. Reducing the number of features reduces storage requirements and training times, while facilitating data visualization and understanding. Additionally, the predictive performance of data models can be improved by defying the curse of dimensionality. It is argued that very high variable correlation does not mean absence of variable complementary. Variable ranking criteria are listed, such as saliency, entropy, smoothness, density and reliability. A scheme for treatment of dirty data is suggested, consisting of detecting outlier examples using top ranking variables.

Dietterich [4] implements ensemble classifiers as a combination of weaker learning algorithms. It argues that individual learning algorithms may achieve poor performance for statistical, computational or representational reasons. Hence, the combination of such learners through the use of bagging, boosting or Bayesian voting results in models with higher performance. The conclusion is that ensemble classifiers may obtain high accuracy by combining less accurate methods.

Qiu *et al.* [5] uses a series of deep belief networks (DBN) as input to an ensemble model. The ensemble model is a support vector regressor (SVR) which aggregates the DBN outputs to make decimal predictions on time series data. The method is found to outperform benchmark methods like standard SVR and ANN on four different datasets.

Deng [6] gives a detailed survey of architectures, algorithms and applications of deep learning. Popular models like deep belief networks, autoencoders, restricted boltzmann machines and recurrent neural networks (RNN) are presented and discussed. It is noted that a fundamental problem with the backpropagation algorithm for deep architectures is that errors become minuscule and ineffective as they are propagated in successive layers. Pre-training of layer weights is suggested as a way of addressing this concern in practice. Choice of deep architecture hyperparameters such as learning rate scheduling, regularization, number of layers and number of units is argued to be very difficult in practice.

Bengio [7] summarizes many important aspects of deep learning and its application to numerous problems. It argues that there is a theoretical limitation on shallow architectures, making them require an exponential number of computational elements compared to deeper models, and that use of shallow models often leads to poor generalization. Deep architectures are said to be suitable when learning complex high-varying functions is required, with little use of human input, and

from very large sets of data. Even so, deep architectures using gradient-based training can often get stuck in local minima or plateaus with worse results than shallow models because their levels close to the input are poorly optimized. Hyper-parameter and parameter optimization in deep architectures is once more noted as very challenging, saying that the price of better generalization is a more difficult optimization problem. Decision trees are noted as poor models for generalizing to new variations, and layer-wise training of deep architectures is suggested.

Hinton and Salakhutdinov [8] argues that gradient descent only works well for the tuning of neural network weights if the initial weights are close to a good solution. Large weights typically find poor local minimas, while small weights make training slow and challenging. Ways of initializing weights in deep autoencoder networks are discussed. Deep autoencoder models for nonlinear dimensionality reduction is suggested as superior to traditional dimensionality reduction methods.

Bengio *et al.* [9] discusses the training of deep architectures. A greedy layer-wise training strategy to optimize the initial weights of a deep belief network is suggested. Deep architectures are highlighted as desirable when representing nonlinear and highly-varying functions, such as in vision, language, speech and robotics.

Bengio and Lecun [10] provides mathematical and empirical evidence suggesting that deep architectures have generalization capabilities that exceed those of shallow architectures. Particularly kernel methods are noted as being fundamentally limited in their ability to learn complex high-dimensional functions. Deep models are said to perform well with regards to computational and statistical efficiency. Three ways of incorporating prior knowledge in machine learning models are suggested, either through the representation of the data, the architecture of the model, or the model parameters like loss functions and regularizers. Once more, layer-wise unsupervised learning follow by gradient descent is suggested to prevent deep architectures from converging to poor local minimas or plateaus.

Krizhevsky *et al.* [11] achieves remarkable success in image classification tasks using deep convolutional neural networks. Extensive data augmentation is used to increase the size of the training set by a factor of 2048. The neural network architecture consists of 60 million parameters, 650.00 neurons in fire layers, and uses features like dropout regularization, ReLU activation, momentum and weight decay. Szegedy *et al.* [12] uses similar architecture for image recognition, while also implementing an algorithm for object detection. Ronneberger *et al.* [13] uses a deep convolution neural network to perform biomedical image segmentation. The architecture consists of a contracting path to capture context and a symmetric expanding path that enables precise localization. Strong use of data augmentation enables to model to be trained on very few images.

Srivastava *et al.* [14] proposes dropout to address overfitting in neural networks. By randomly dropping units during training, co-adaption of nearby units is avoided. At test time, a single untinned network is used with smaller weights to gain an averaging of the predictions. Through this technique, major improvements over

other existing regularization methods are found, noted by improved performance for tasks in vision, speech recognition, document classification and computational biology. Dropout can also be used during prediction, adding some degree of modelling uncertainty to the predictions.

Gal and Ghahramani [15] analyses the use of dropout regularization as a Bayesian approximation to represent the model uncertainty of deep learning models. It is emphasised that standard deep learning models for regression and classification do not capture model uncertainty, and proven that neural networks with arbitrary depth and nonlinearities using dropout regularization are mathematically equivalent to probabilistic deep Gaussian processes. The model uncertainty may be obtained by making multiple predictions with the same model, for which the output will be approximately Gaussian distributed given a sufficient number of predictions. Using this uncertainty, traditional Bayesian maintenance models can be applied.

2.3 Predictive maintenance and anomaly detection

Ileby and Knutsen [16] argues that costs can be reduced while safety can be improved in the oil and gas industry through the use of advanced condition monitoring technologies. The Ivar Aasen offshore platform operated by Aker BP is used as an example, for which performance metrics using real-time and historical condition monitoring are implemented. Such monitoring is found to be essential in expanding margins by reducing productions costs, and achieving low break-even production costs. It is concluded that well-planned, timely maintenance is preferred as opposed to reactive maintenance.

Shin and Jun [17] lists important aspects of condition based maintenance (CBM), namely its ability to predict degradation and perform repairs ahead of failure. Data-driven, model-based and hybrid approaches are highlighted, and a procedure for CBM is defined as consisting of data processing, diagnostics, prognostics and execution of maintenance operation. Parameters to be monitored, inspection frequency and warning limits must be defined at the decision making level.

Görnitz *et al.* [18] suggests the use of semi-supervised anomaly detection, in which a small amount of labeled data is used together with a large amount of unlabeled data. Such models may detect anomalies by finding a concise description of the normal data, by comparing the labeled and unlabeled samples e.g. through the use of clustering methods. Semi-supervised methods allow for the inclusion of prior and expert knowledge in anomaly detection systems where applicable.

Ahmad *et al.* [19] defines anomalies as points in time where the behavior of a system is unusual and significantly different from previous normal behavior, and aims to find such anomalies in a real-time high velocity environment through unsupervised methods. It is noted that anomaly detection models need to be adaptable as many systems experience concept drift[20], meaning changes in the

configuration over time may alter the behavior of the system and thus also changing what would be considered normal behavior. An important balance between early detection and false positives is discussed.

Chandola *et al.* [21] gives a structured and comprehensive overview of the research on anomaly detection, which it defines as finding patterns in data that do not conform to expected behavior. The challenge of defining the normal region is noted, highlighting that this region should encompass every possible area of normal behavior and can be continuously evolving. Defining labeled data for training and validation of anomaly detection models is said to be a major issue in practice. Different types of anomalies are explained, such as point anomalies, contextual anomalies and collective anomalies. Scoring and classification of anomalies is discussed, as well as possible methods such as nearest neighbor-based, clustering-based and statistical methods.

Zhao *et al.* [22] suggests using autoencoder networks with restricted boltzmann machines to predict early failures in wind turbines. The network is trained using unlabeled data. Extreme values theory is utilizes to define adaptive thresholds to reduce the number of false alarms and enhance predictive capabilities.

Hill and Minsker [23] implements anomaly detection for an autoregressive model of wind speed data. An anomaly is defined as a measurement deviating significantly from its one-step-ahead prediction. A constant threshold is used, found through the use of performance indicators. Various models such as nearest neighbor, clustering, neural networks and decision trees are compared, for which multilayer perceptron (MLP) models are found to have the highest performance. Different reasons are given for the presence of anomalies, namely sensor errors, data transmission errors or abnormal system behavior.

Wu *et al.* [24] implements degradation monitoring using a key performance indicator (KPI) for heat exchangers operating in a batch process environment. The KPI is found through data-driven modelling techniques based on partial least squares. Because the degradation increases in batches, a periodic pattern of degradation is found. This is then used to optimize maintenance planning to great effect.

Tang and Fishwick [25] presents an early study on the use of feedforward neural networks as models for time series forecasting. Improved performance is found for time series with short memory in particular. Multiple-step-ahead forecasting is suggested for long-term forecasting models. It is concluded that neural networks can be a promising alternative for time series forecasting. An interesting implementation detail of using direct connections from input to output units is included, allowing for a strictly linear relation.

Jiang *et al.* [26] looks to diagnose rolling bearings using deep recurrent neural networks. An adaptive learning rate to improve training performance is suggested. Deep recurrent neural networks are found to be more efficient than traditional intelligent fault diagnosis methods at capturing long-term dependencies.

Malhotra *et al.* [27] uses the reconstruction error of an LSTM-based encoder-decoder model to compute anomaly likelihood. The model is trained on instances of normal time series, with the target being the time series itself. The resulting model, named EncDec-AD, is found to perform well both for predictable and unpredictable time series.

Park *et al.* [28] implements a lightweight, real-time fault detection on a single-board computer. A Raspberry Pi is used, connected to certain sensors and relevant cloud computing platforms. Edge computing is utilized to reduce the workload of the concentrated center cloud. Such miniature systems, with limited storage and computing capabilities, are argued to be suitable in Smart Factory settings. The fault model uses LSTM recurrent neural networks, and techniques such as early stopping and dropout regularization, as well as the Adam optimizer and ReLU activation function.

Khan and Yairi [29] presents a detailed and systematic review of machine learning models used for condition monitoring and degradation modelling, with a particular focus on deep learning methods. Some challenges of system diagnostics are highlighted, namely identification of faults, low-dimensional visualization, robust separation of faults, handling of noisy sensor readings, and modelling of physical processes and health degradation trends. Several strengths and limitations of various models are listed. It is argued that deep learning is not as easily applied to condition monitoring as to other disciplines like image processing and speech recognition, because faults and failures can be heterogeneous and vary according to different environments. Recommendations for future work in deep learning are noted, namely minimizing complexity to enhance explainability, factor in uncertainties, define selection criterion for applicable models, and incorporating expert knowledge.

Yang *et al.* [30] proposes a random forest algorithm for machine fault diagnosis. A number of decision trees are used to improve a single tree classifier. Benefits of tree based models are highlighted as fast execution speed and high performance. Their findings suggest the random forest algorithm may outperform traditional support vector machine and classification tree models.

2.4 Heat exchanger fouling

Kumra *et al.* [31] attempts to predict the heat transfer rate of a wire-on-tube type heat exchanger using support vector machine (SVM) and neural network models. It appears that a very limited number of data samples are used, 14 for training and 5 for testing. Additionally, a large number of input parameters are used.

Ardsomang *et al.* [32] uses an autoassociative kernel regressor to predict the effect of fouling on heat transfer resistance, and a general path model (GPM) to perform remaining useful life (RUL) predictions using the output of the regression model. A test rig is used for which accurate measurements are available, and fouling is

added by adding Kaolin clay to the water on the tube side of the heat exchanger. The regression model is able to predict the level of fouling well by calculating the LMTD. Cooling rates are kept constant and all available temperature and flow measurements are used in the predictions.

Lalot and Pálsson [33] implements a neural network to detect levels of fouling in cross-flow heat exchangers. The data is generated through the use of numerical models representing clean and fouled heat exchangers, from which time series are generated by simulations. A shallow network architecture of 4 nonlinear units and one linear unit is used. Fouling is introduced by progressively decreasing the overall convection heat transfer coefficient during the simulations. Temperature and flow measurements are used in the clean and fouled heat exchanger models to determine the fouling factor.

Jerónimo *et al.* [34] monitors the thermal efficiency of fouled heat exchangers using thermodynamic methods like number of transfer units (NTU) and thermal efficiency. Tests are performed on data from an oil refinery, for which it is noted that the flow rates and physical properties may change over time. These readings may also differ from the design conditions, meaning day-to-day measurement of thermal efficiency cannot be easily compared to the predicted efficiency based on the design calculations. It is argued that changes in physical properties does not affect the accuracy of the models significantly. Efficiency is calculated by the use of temperature and flow readings, and compared to estimated efficiencies for the clean and fouled heat exchanger state. The methods are said to yield fairly good results in the assessment of fouling conditions.

Liporace and Oliveira [35] looks at the effects of fouling in the preheat train of a distillation column for crude oil treatment. This unit is chosen due to high levels of asphaltene in the oil causing fouling in the heat exchangers. Added fuel consumption due to fouling in the distillation column reboiler is estimated to cost 1.8mill USD per year excluding lost revenue from loss of unit throughput for this unit alone, indicating that the presence of fouling can have significant economical consequences.

2.5 Summary

Previous work at Equinor suggests the use of both unsupervised and supervised machine learning methods to identify heat exchanger fouling, although without suggesting specific fouling indicators using such methods. Study of related work argue in favor of LSTM models for analysis of multivariate time series data. Use of deep architectures is advised when fitting complex high-varying functions using very large datasets. Feature selection, for instance through variable ranking, is recommended to reduce the number of features. Pre-training of layers is suggested to avoid the vanishing gradient problem in the backpropagation algorithm. Hyperparameters such as the Adam optimizer and ReLU activation function are re-

commended, although hyperparameter selection in general is noted as very challenging. Exceptional performance for deep neural networks is shown for tasks like image classification, with extensive use of data augmentation to increase the dataset size. Dropout regularization is found to have major improvements over other existing regularization methods. Advanced schemes like weight initialization, learning rate scheduling and layer-wise training are proposed.

Condition-based maintenance and real-time anomaly detection is argued to have major financial impact in industries. Several unsupervised, semi-supervised and supervised anomaly detection techniques are suggested. Autoencoders, random forest, neural network and LSTM models are implemented for different domains. Deep learning is noted as more difficult to implement for condition monitoring than for other disciplines, because the underlying systems are very different, and that explainability must often be sacrificed in favor of model complexity.

Neural networks are implemented for fouling detection in heat exchangers, although with very restricted model architectures and few data samples. It is noted that changes in operating conditions over time make fouling prediction by the use of thermal efficiency difficult, although limited effects from changes in physical properties are found. Monitoring in practice is said to vary for each facility, and lack of necessary measurements is noted as a major obstacle in the use of traditional monitoring techniques. Seemingly, the application of machine learning methods for industrial systems is still a work in progress, despite the remarkable success of such methods, and deep learning in particular, in other domains. For heat exchanger systems and fouling specifically, limited research is found using large datasets and deep architectures.

Chapter 3

Theoretical framework

This chapter provides insight into the two primary domains of this thesis, namely petroleum processing with a focus on heat exchangers and machine learning for predictive maintenance in industrial systems. Due to the high degree of interdisciplinarity, each topic is covered such that readers with elementary knowledge in either field can understand the thesis as a whole. Other domains such as maintenance theory and statistics are briefly mentioned, but not explained in detail nor considered essential. Chemistry, thermodynamics and petroleum theory is covered to an adequate level for understanding heat exchanger fundamentals. Understanding the challenges related to traditional monitoring and maintenance of heat exchangers is essential in applying and evaluating the performance of machine learning algorithms to this domain. Machine learning theory is presented more thoroughly so that readers who are not familiar with this topic can grasp the most crucial aspects prior to reading about the thesis methodology and findings.

3.1 Oil and gas processing

In this section, fundamental theory concerning oil and gas processing, pipe liquid flow and heat transfer is presented. The content is based on books by Campbell [36][37] and Bergman *et al.* [38], and the thesis by Jensen and Nordhus [1]. Individual references are included where deemed appropriate.

3.1.1 The oil and gas value chain

The petroleum industry operate many complex, large-scale industrial processes and technologies. Producing hydrocarbon products with widespread use like natural gas, fuel and more in an optimized manner requires thoughtful treatment in diverse stages. Oil and gas reservoirs, either onshore or offshore, are located through the use of geophysical imaging technologies. Reservoirs often contain a mixture of gas, crude oil, water and other minerals. This mixture is extracted

using an oil or gas well, leading a wellstream through a series of pipes and processing equipment. Treatment of the wellstream is complex, involving transport of multiphase flow, several stages of separation, removal of carbon dioxide, gas dehydration, stabilization of crude oil, transportation, onshore processing and more.

An important step of the wellstream processing is to separate the various hydrocarbons. This is done to derive products which can be transported, sold and used individually. Some of these products are as follows:

- Natural gas, and potentially liquid natural gas (LNG) if cooled, for home and industrial applications such as heating and cooking. Natural gas contains mainly methane, with trace amounts of heavier hydrocarbons as well as compounds such as CO_2 and nitrogen.
- Ethane, which is used to produce ethylene for the chemical industry, e.g. for production of plastic.
- Liquefied petroleum gases (LPG) such as propane, butane or a mixture of these, typically used as propellants.
- Heavier hydrocarbons commonly used in gasoline, such as pentane, hexane, heptane and octane.
- Even heavier hydrocarbons used in production of jet fuel, diesel, asphalt and more.

The separation process happens in multiple phases. An overview of this process is presented in Figure 3.1, typically performed at a facility relatively close to the well such as a drilling platform or offshore processing plant. Water is separated from gas and oil, and either cleaned to the point where it can be released back into the ocean, or pumped back into the well to maintain the reservoir pressure. Lighter carbons such as methane, ethane, propane and butane are separated and transferred for further processing in gas form. The heavier hydrocarbons are exported in liquid form as stabilized oil. Stabilized oil, as opposed to crude oil from the reservoir, contains little hydrogen sulfide and has a reduced vapor pressure, reducing the volatility and making it suitable for shipping by oil tankers[36].

During processing, some or all contents of compounds like water, CO_2, N_2, O_2 and Hg must be removed. This may be done based on product requirements or to reduce environmental impact. Removal of H_20 is particularly important for transfer of water-hydrocarbon systems, because the water may form hydrates or ice which hinders transport and reduces equipment performance. Hydrates are crystallized water-based solids somewhat similar to ice. Hydrocarbon fluids are dehydrated in order to avoid the formation of hydrates and ice, and to prevent corrosion[37].

A possible subsystem of a processing facility is seen in Figure 3.2. A largely gaseous fluid of hydrocarbons is transported through a series of pipes before entering a heat exchanger. The objective of the heat exchanger is to reduce the temperature of the fluid, causing heavier compounds to go from gas to liquid form. In the sep-

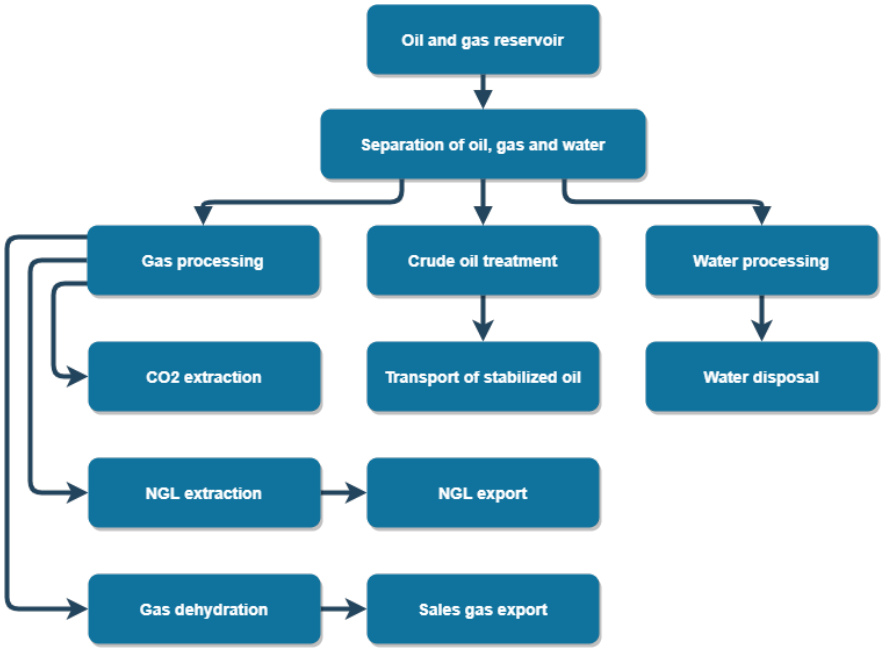


Figure 3.1: An overview of the extraction and initial processing of oil and gas from a reservoir.

arator, gas will typically rise while liquids fall downwards, enabling the separation of these components. One of the fluids, for instance the gas fluid as indicated on the illustration, is extracted from the separator and used as a coolant for the heat exchanger while the liquid flow is extracted elsewhere. Other coolants may also be used. Each fluid may be processed further before they eventually are exported for sale.

The hydrocarbons are processed further, for instance at an onshore facility, deriving products exported to different markets. Extraction of the light compounds, up to C_9, is called natural gas liquids (NGL) extraction. This is typically performed either for the natural gas export to meet its sales specifications or to increase the market value of the overall petroleum export[36]. For gases, the sales specifications may include the heating value of the product, sulfur content, maximum delivery temperature, water content, hydrocarbon dew point and others. Sales specifications for liquids vary based on the product, but typically some restrictions on CO_2, C_1 and C_2 content are imposed.

3.1.2 Introduction to hydrocarbon chemistry

Oil and gas are hydrocarbons formed through geochemical processes, stemming from fossilized organic materials. Knowledge of the various hydrocarbon compounds is vital in ensuring safe and efficient extraction of this matter from reservoirs, especially regarding properties like phase, density, conductivity and more.

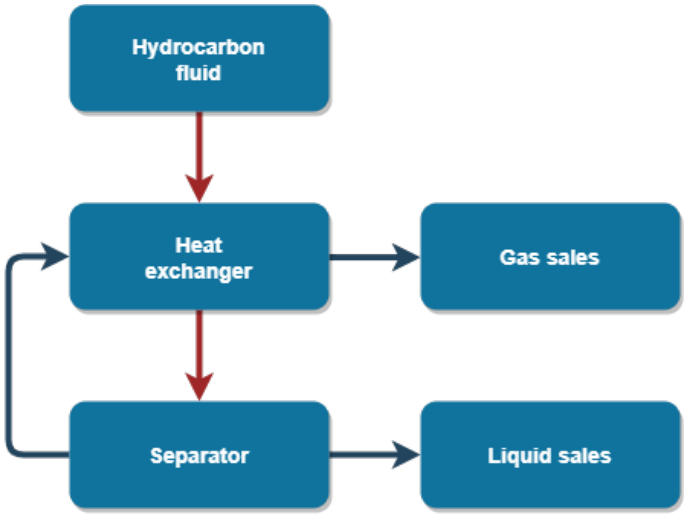


Figure 3.2: Example of a subsystem in oil and gas processing. A warm gas is cooled in a heat exchanger, before entering a separator to extract gas and liquid components. The gas outlet of the separator is used as the coolant.

Some challenges experienced in practice are listed below to demonstrate why[37]:

- When pumping or transporting liquids, it is important to always operate so that vapor formation and cavitation can be avoided.
- To separate various compounds using a separator, operating temperatures and pressures must be set so that the correct separation may occur. This means the boiling point of whichever compounds is to be extracted as gas must have been reached, while remaining below the boiling point of those compounds that are to be extracted as liquids.
- The contents of the wellstream may change over time, requiring adjustments in fluid calculations.
- As compounds are cooled, liquids or solids may form. This can cause buildup of material on the pipe or heat exchanger walls.
- Lack of knowledge regarding the optimal operating point may lead to wasteful operating temperatures and pressures.
- Multiphase flow in general makes calculations and estimations difficult.

Generally, a single compound can be in either solid, liquid or gas form, depending on its molecular kinetic energy. Solids have relatively low kinetic energy, hence the molecules vibrate in place rather than moving around like in a liquid or gas. Changing the temperature, specific volume or pressure of a compound changes the kinetic energy. The physical properties and phase of the compound may change according to these same variables. The state variables of a compound are related through equations of state (EoS). The equations may be used to calculate properties like density, vapor pressure and more, and take the general form of Eq.

Table 3.1: Symbols used in equations of state and mixing rules

Symbol	Definition	Unit
p	Pressure	Pa
V	Volume	m^3
V_m	Molar volume	m^3/mol
n	Number of moles	$-$
R_g	Universal gas constant	$J/K \cdot mol$
T	Absolute temperature	K
k_B	Boltzmann constant	J/K
N_A	Avogadro constant	mol^{-1}

3.1. The function f may be found through empirical calculations, and thus the accuracy will vary based on input parameters and the model complexity.

A simple equation of state is the general gas equation or ideal gas law seen in Eq. 3.2. It is roughly accurate for low pressures and moderate temperatures. In oil and gas processing, however, more accurate equations of state are desired. Equations like the Soave-Redlich-Kwong EoS (SRK EoS) and Peng-Robinson EoS (PR EoS) are often used, as seen in Eq. 3.3 and Eq. 3.4. Both SRK EoS and PR EoS contain parameters such as a, b and α, which empirically estimated functions of the remaining parameters and the fluid composition. For Peng-Robinson, these are seen in Eq. A.1, A.2, A.3, A.4 and A.5. Symbols used in the equations of state are explained in Table 3.1.

$$f(p, V, T) = 0 \tag{3.1}$$

$$pV = nRT = nk_B N_A T \tag{3.2}$$

$$p = \frac{RT}{V_m - b} - \frac{a\alpha}{V_m(V_m + b)} \tag{3.3}$$

$$p = \frac{RT}{V_m - b} - \frac{a\alpha}{V_m^2 + 2bV_m - b^2} \tag{3.4}$$

In fluids consisting of several compounds, mixing rules must be applied in order to estimate how the properties of each component affect the total mixture. The mixing rule defines how a and b are calculated in the equations of state. An example is the van der Waals one-fluid mixing rule seen in Eq. 3.5 and Eq. 3.6. a_{ij} and b_i are the interaction parameters for the different compounds. For this thesis it is sufficient to be aware that for multiphase and multicomponent flow, the choice of equation of state and mixing rule can affect the result significantly.

$$a_{mix} = \sum_{i=1}\sum_{j=1} y_i y_j a_{ij} \tag{3.5}$$

$$b_{mix} = \sum_{i=1} y_i b_i \tag{3.6}$$

In separating the various hydrocarbon products, knowledge of the fluid properties and phase behavior plays a vital part. Phase behavior refers to the behavior of a compound when exposed to different temperatures, pressures and specific volumes. Using these three parameters, a three-dimensional diagram showing the compound behaviour can be plotted. Because such three dimensional figures are difficult to use in practice, pressure-temperature (PT) or pressure-volume (PV) two-dimensional plots are often used instead[36]. The fluid properties and phase behavior must be calculated or approximated when performing thermodynamic calculations for oil and gas systems.

An example of a P-T diagram can be seen in figure 3.3. Solid, liquid and vapor phases are clearly visible. Additionally, a rectangle labeled supercritical fluid can be seen. Here, the compound has the appearance of a gas, while possessing other properties than in the regular vapor phase. The behavior of the compound is each phase dictates how calculations have to be made. For instance, while in the supercritical phase, the density is higher than in the regular vapor phase, however the compressibility is greater than that of a regular liquid. These properties must be accounted for when performing calculations on the fluid.

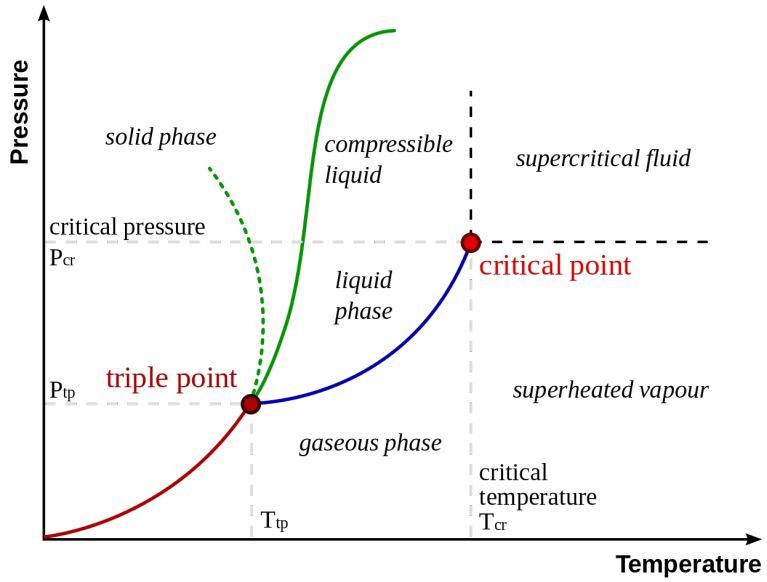


Figure 3.3: Example of a Pressure-Temperature phase diagram [39]

In a mixture of compounds, the same parameters as before apply, in addition to the mixture composition. The phase diagram equivalent for this mixture can be represented as a phase envelope. This diagram is commonly used to describe in which operational areas liquids or solids are found for the mixture. An example is seen in Figure 3.4. Within the phase envelope, the mixture will be be a multiphase flow, consisting of gases, liquids and potentially solids. Notice the retrograde region, in which the mixture experience behavior opposite of what is considered normal, by condensing when the pressure is lowered or temperature is increased. The dotted lines indicate a constant vapor-liquid ratio. The vapor-liquid ratio may be found by performing a equilibrium-flash calculation.

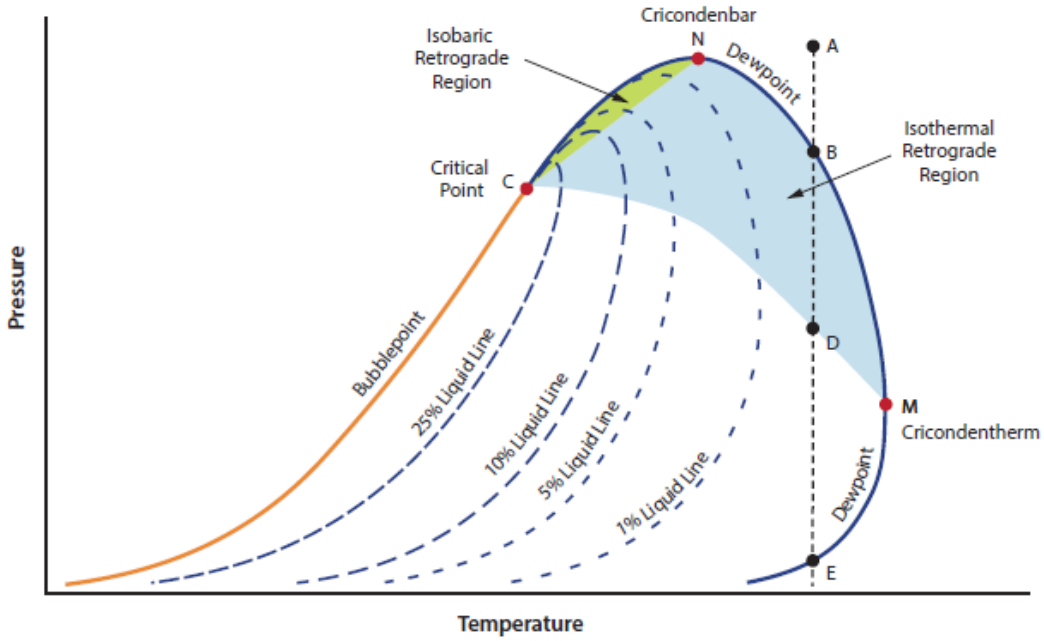


Figure 3.4: Example of a Pressure-Temperature phase envelope for a mixture of compounds [36]

When calculating phase behavior of multi-compound flows, it is important to know the content of each compound. Generally, compounds up to contents of C_{7+} must be accounted for in gas flow[36]. This is because small contents of each heavier compound and droplets are present in the gas. Similarly for crude oil, known contents up to C_{20+} may be required[36]. Composition effects the shape of the phase envelope significantly, and thus must be accurately measured in order to perform reliable phase behavior calculations. For instance, the dew point line of lean natural gas compositions are greatly affected by the contents of heavier carbons such as C_{7+}, CO_2 and other components[37].

3.1.3 Processing equipment

Some essential processing equipment is listed below, before a brief overview of a joined system is given[37].

Separators and distillation columns are used to separate gases and liquids. Liquids are typically pulled downwards by gravity while the gas escapes at the top of the separator. Such equipment is designed with several requirements in mind, some of which are sufficient section area for the liquid and gas to separate, sufficient liquid capacity, and sufficient height or length to allow droplets in the gas to settle. Additional separation may be required to remove carryover fluids. Filters are used to clean the gas of liquid or solid particles carried by the gas through the separator. To ensure the desired separation takes places, it is important to control the operating point of the flow entering the separator. Here, analysis of the flow phase envelope and the phase diagrams of each compound proves useful.

Separators may also separate three different phases rather than just gas and liquid. In this case, two different liquids are separated, for instance oil and water. Such separators are designed to provide sufficient separation of the liquids, for example to meet a certain oil percentage in the separated water. Separators alone are not capable of sufficiently separating the various hydrocarbon products. For this purpose, distillation columns are used. In essence, these are large, vertical tanks in which different operating conditions are ensured throughout the tank so that specific compounds can be extracted at each level.

Compressors and pumps increase pressure by reducing volume. Knowing that the compressibility of liquids is very low compared to that of gases, it is often desirable to assure a sufficiently low liquid fraction for the flow entering a compressor. Compressors are used to ensure a desired pressure is reached when processing the wellstream components, or simply to increase pressure for transportation. When handling a multiphase flow, pumps can be used for this same purpose.

Expanders reduce pressure and temperature by increasing volume. This can for instance be done to a flow before entering a distillation column, ensuring the desired operating temperature and pressure is reached. Expanders and compressors can be connected as part of a turboexpander, where the work produced by the expanding gas is used as energy resource for the compressor.

Heat Exchangers are used to cool one fluid abd heat another by exchanging heat between the two, while keeping the flows separate. Typically, flows are arranged either so that they enter the exchanger in parallel or opposite of each other. The latter ensures the most efficient heat transfer due to increased temperature difference between the fluids along the exchanger surface. Different designs exist, such as shell-and-tube, plate-and-frame, pipe-in-pipe and others. Performance is typically measured by calculating the heat transfer coefficient between the fluids. As no energy is accumulated in the heat exchanger itself, all increase in energy of the cold side system should equal the decrease in energy of the warm side sys-

tem minus energy lost to the surroundings, which is typically rather low. Heat exchangers are discussed in greater detail in Chapter 3.2.

A system example containing the previously discussed componenets can be seen in figure 3.5. This exact process is used to extract heavier carbon products to be treated as liquids from methane, in what is known as a demethanizer. An inlet stream with a relatively high pressure and temperature is cooled by a heat exchanger, before entering a first-stage gas-liquid separator. The pressure and temperature of the gas flow is reduced further by a turboexpander, while the same is done for the liquid flow using a valve. Both flows enter the demethanizer, in which almost pure methane leaves the top of the column. This gas is heated, compressed and exported in a pipeline. The liquid products are treated in the distillation column, pumped, cooled and exported.

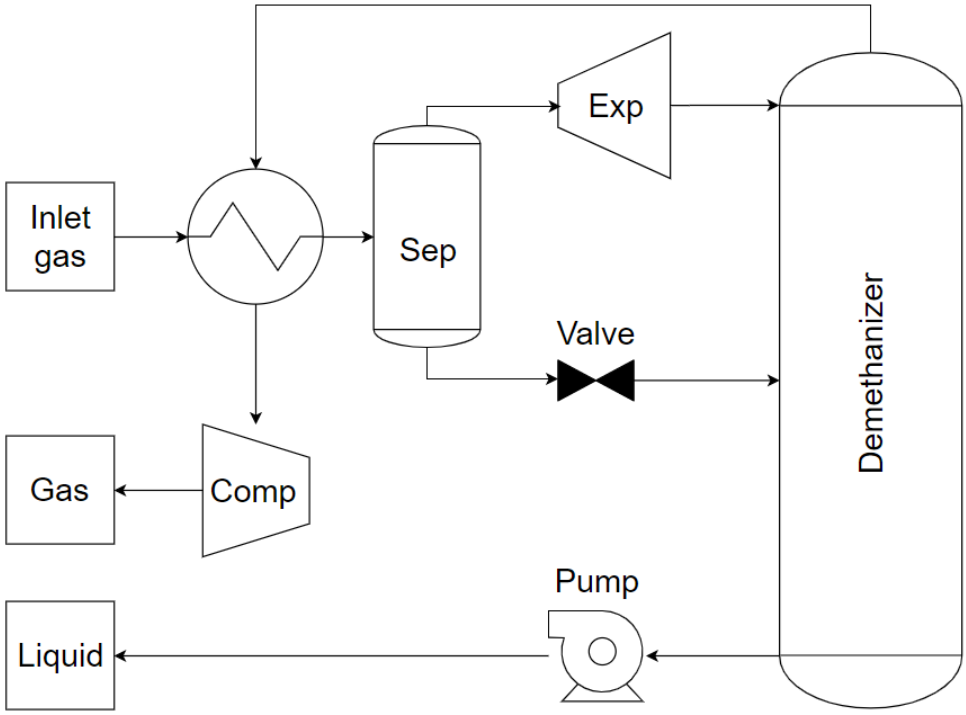


Figure 3.5: Simplified overview of more complex oil and gas processing using a demethanizer. Note that a real distillation column contains more flow outlets at various levels of the column than is illustrated here.

3.1.4 Pipe flow

Fluids that flow through pipes or processing equipment experience drops in pressure. This occurs due to several reasons, such as friction between the fluid and pipe, friction in bends and valves, or changes in pipe elevation. Pressure can also be increased by the use of pumps or compressors, or decreased using valves.

Table 3.2: Symbols used in pipe flow equations

Symbol	Definition	Unit
Δp	Pressure difference	Pa
L	Length of pipe	m
f_D	Darcy friction factor	$-$
ρ	Fluid density	kg/m^3
v	Mean flow velocity	m/s
D	Hydraulic diameter	m

An equation relating pressure loss and fluid characteristics for regular pipe flow is the Darcy-Weisbach equation. It expresses the pressure loss per unit of length:

$$\frac{\Delta p}{L} = f_D \frac{\rho v^2}{2D} \tag{3.7}$$

Parameters inserted into the Darcy-Weisbach equation depend on flow conditions, fluid properties and pipe geometry. They are described in Table 3.2. For more advanced pipes or flow geometries, contributions from various sources such as bends, valves, inlets and outlets must be accounted for. Additionally, a two-phase friction multiplier has to be introduced to account for two-phase flow conditions[1]. This leads to drastically more advanced formulas. For this thesis, it is sufficient to notice that despite the introduction of more advanced terms, the pressure drop will remain a quadratic function of the fluid velocity, as indicated by the Darcy-Weisbach formula. The velocity is affected by the mass flow, fluid density and the cross section area of the flow pipe. Thus, these properties along with the fluid viscosity and pipe arrangements will determine the total pressure loss of the fluid. Defining an accurate expression for the pressure loss in practice may be very difficult, depending on the pipe configurations.

3.1.5 Heat transfer fundamentals

When two substances at different temperatures are exposed to each other, thermal energy will be transported from the warm substance to the cold substance. In an oil and gas process, these substances may be a warm hydrocarbon mixture and a cold flow of some coolant. If a certain gas temperature is required at any point, for instance to separate methane gas from heavier hydrocarbons in a separator, the temperature of the gas must be reduced prior to the separation process. In doing so, heat is transferred from the gas stream to the coolant. The most commonly used coolants are seawater or water mixed with an antifreeze like ethylene glycol[36]. The antifreeze is added so that the freezing point is lowered and more efficient heat transfer can take place while avoiding solids forming in the coolant flow.

Table 3.3: Symbols used in heat transfer equations

Symbol	Definition	Unit
q	Heat	W
k	Thermal conductivity	W/mK
A	Heat transfer area	m^2
$q_x^{''}$	Rate of heat transfer per unit area	W/m^2
dT	Temperature gradient	K
dx	Length of transfer	m
h	Convection heat transfer coefficient	W/m^2K
T_s	Surface temperature	K
T_∞	Fluid bulk temperature	K
ϵ_{rad}	Radiative efficiency	-
σ_b	Stefan Boltzmann constant	W/m^2K^4
T_{sur}	Surrounding temperature	K
U	Overall heat transfer coefficient	W/m^2K
R	Thermal resistance	m^2K/W

Because the process flow and the coolant must remain separated, they cannot be physically exposed to each other and transfer thermal energy directly. Instead, they are transported along each other with a solid wall acting as a physical barrier. This barrier should ideally have high thermal conductivity for the transfer of heat to exchange as freely as possible between the two fluids[38]. Devices invented for such purposes are called heat exchangers, and are discussed in Chapter 3.2.

Heat transfer can occur through different mechanisms. They are summarized as follows. Symbols used in the equations are explained in Table 3.3.

- **Conduction** is diffusion of energy due to random molecular motion, described by Eq. 3.8
- **Convection** is heat transfer between fluids in motion on each side of a bounding surface, described by Eq. 3.9
- **Thermal Radiation** is energy emitted by a matter at a given temperature, described by Eq. 3.10 if simplifications are made for which the transfer of energy happens between a small and much larger surface

$$q_x^{''} = \frac{q_x}{A} = -k \cdot \frac{dT}{dx} \tag{3.8}$$

$$q^{''} = \frac{q}{A} = h \cdot (T_s - T_\infty) \tag{3.9}$$

$$q^{''} = \epsilon_{rad} \cdot \sigma_b \cdot (T_s^4 - T_{sur}^4) \tag{3.10}$$

For any heat transfer problem, the transfer heat q may be found by calculating the energy balance. One or more of the previous mentioned types of heat transfer will contribute in these equations. For instance, heat transfer between two fluids separated by a plane wall in a steady state situation will contain convective terms for each of the flows and a conductive term for the wall itself. The expressions inserted for each parameter in equations 3.8, 3.9 and 3.10 will change depending on the heat transfer situation. Heat transfer through a plane wall will have different expressions for thermal conductivity than through a pipe wall.

Once an expression for q has been derived, other useful properties like thermal resistance R and overall heat transfer coefficient U can be defined. The equations for U and R will therefore also depend on the heat exchange situation, for instance through a plane wall or a pipe as mentioned. R relates the heat transfer rate to the temperature difference of the fluids by the following equation:

$$q = q'' \cdot A = \frac{dT}{R} \tag{3.11}$$

U describes how heat flows through a system. The heat transfer coefficient is often used in practice to measure heat transfer performance. It can be calculated for heat transfer through a plane wall as follows:

$$U = \frac{q}{A \cdot (T_{\infty,1} - T_{\infty,2})} = \frac{q}{A \cdot dT} \tag{3.12}$$

The total thermal resistance for the same heat transfer situation can be calculated as follows:

$$R = R_{tot} = R_h + R_{wall} + R_c = \frac{1}{(hA)_h} + \frac{\Delta x}{k} + \frac{1}{(hA)_c} \tag{3.13}$$

From the total thermal resistance, Eq. 3.12 for the overall heat transfer coefficient can be rewritten to the following:

$$\frac{1}{UA} = R_{tot} = \frac{1}{(hA)_h} + \frac{\Delta x}{k} + \frac{1}{(hA)_c} \tag{3.14}$$

Additional conditions may affect heat transfer, such as the presence of unwanted material on the heat transfer surface, called fouling. Fouling and what may cause fouling is discussed further in Chapter 3.2. If fouling is present on the heat transfer surfaces, additional terms must be introduced to account for the increase in thermal resistance as follows:

$$R = R_{tot} = R_h + R_{f,h} + R_{wall} + R_{f,c} + R_c \tag{3.15}$$

The overall heat transfer coefficient for heat transfer between two fluids through a plane wall can now be expressed for the case where fouling is present as follows:

$$\frac{1}{UA} = R_{tot} = \frac{1}{(hA)_h} + R_{f,h} + \frac{\Delta x}{k} + \frac{1}{(hA)_c} + R_{f,c} \tag{3.16}$$

If overall heat transfer coefficient U is known for heat transfer between two fluids through a clean surface and U_f is known for heat transfer of the same fluids through a fouled surface, the fouling factor R_f may be found as follows:

$$R_f = \frac{1}{U_f} - \frac{1}{U} \tag{3.17}$$

The overall heat transfer coefficient, total thermal resistance or fouling factor can be used as a measure of heat transfer efficiency. In practice, calculating either of these properties can be very challenging. This is especially true for multiphase flow, for which effects like condensation in the fluid must be accounted for[38]. Values like thermal conductivity k and convection heat transfer coefficient h must be known or estimated for each fluid or surface. These are often listed in tables for specific substance mixtures, or estimated empirically.

3.2 Heat exchangers and fouling

In this section, necessary theory on heat exchangers and fouling mechanisms is presented. Additionally, condition monitoring and maintenance techniques for such equipment are discussed briefly. The content is based on books and papers by Bott [40], Bott and Melo [41] and Shah and Sekulic [42], and the thesis by Jensen and Nordhus [1]. Individual references are included where deemed appropriate.

3.2.1 Heat exchanger design

As mentioned, a physical barrier with high thermal conductivity should separate the fluids during heat transfer. Such a device is called a heat exchanger. Under steady state conditions, very little energy is lost to the surroundings and no energy is stored inside the exchanger itself. This means almost all thermal energy transferred from the hot substance is utilized to heat the cold substance[1].

There are several types of heat exchangers and different flow patterns, which may be chosen for different purposes. Some typical flow patterns are parallel, counterflow and more advanced patterns. Type and flow pattern is chosen according to a set of specifications, such as pressure limits, thermal performance, temperature ranges, flow capacity, maintainability and material selectionBott [40][42]. A brief summary of some popular designs is given below.

- **Shell-and-tube heat exchangers** (STHE) have a very simple design of an encapsulating shell and interior tubes. One fluid flows through the tubes while the other fluid flows through the shell itself, surrounding the tubes. Baffles, in the form of obstructing vanes or panels, are installed to direct the flow and support the tubes. Shell-and-tube is typically chosen for high-pressure or high-temperature conditions due to its robustness. Design features include tube diameter, tube thickness, tube length and tube layout.
- **Plate heat exchangers** (PHE) are built using thin plates of high conductance material, creating a large heat transfer area. Due to the highly limited fluid passage, the flow becomes increasingly turbulent which enhances heat transfer. Plate exchangers are typically more dense than shell-and-tube exchangers, although at the cost of a higher pressure drop. Plate heat exchangers with thin plates may not be usable for high pressure systems.
- **Printed circuit heat exchanger** (PCHE) are somewhat similar to traditional plate exchangers, with even more closely aligned plates to increase the density while also increasing the conductive capabilities. The plates are welded together in stacks, with very small diameter semi-circular holes. Because the density is greater, these exchangers may be more prone to fouling. Unlike the usual plate exchanger, PCHE can usually operate at high pressures.

3.2.2 Operation of heat exchangers

The heat exchangers analyzed in this thesis are used in systems as illustrated in Figure 3.6. A separate coolant system is used to cool a fluid of hydrocarbons, with additional processing both before and after the heat exchanger. The coolant may be part of an open-loop or closed-loop coolant system. In closed-loop systems, the coolant is reused after being recooled in a separate system, such as with fresh water and antifreeze. If antifreeze is used, the coolant is typically recycled rather than disposed. In open-loop systems, the coolant is disposed, as for traditional seawater cooling. Counterflow or more advanced flow patterns are normally used.

The hydrocarbon fluid outlet temperature $T_{h,o}$ is specified to be maintained at a certain level. The valve opening of the coolant channel is controlled according to desired $T_{h,o}$ using a control system. The valve opening is indicated by an opening level between 0 and 100. Ideally, the coolant flow is determined by the valve opening and heat exchanger upstream pressure. However, other factors may apply which causes the coolant flow not to correlate with the valve opening. For example, the coolant resource may be shared with other components and thus limited in supply. Other operating conditions, such as pressures and inlet temperatures, are typically controlled by the operating conditions of upstream and downstream equipment. Ideally, these remain rather constant. Coolant inlet temperature varies according to seawater temperature when this is used as coolant. Thus, as process flow increases or heat exchanger performance decreases, the coolant flow must be increased to maintain the same desired process outlet temperature.

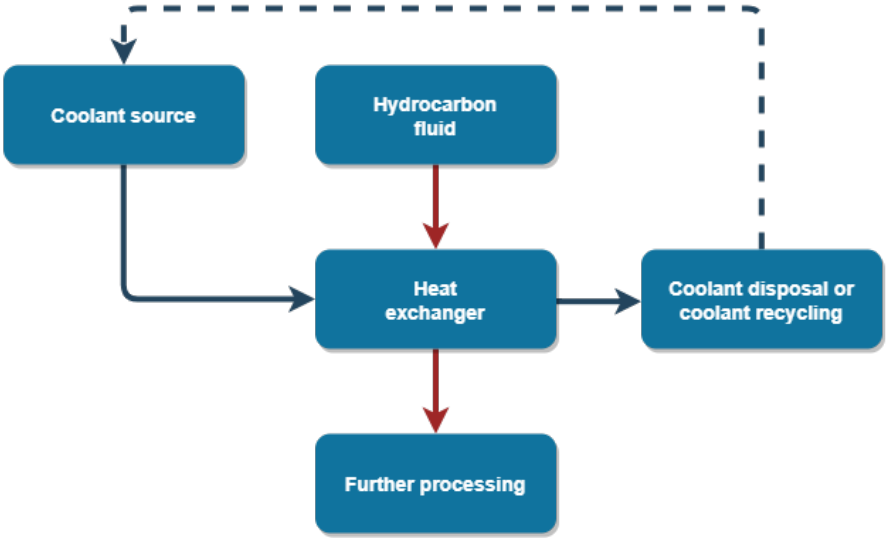


Figure 3.6: Example of heat exchanger system with coolant source.

3.2.3 Fouling in heat exchangers

Because efficient heat transfer requires large surface area while the size of processing equipment must be limited due to facility constraints, increasing heat exchanger density while maintaining heat transfer capabilities is desired[42]. As density increases, the size of the flow channels decrease, and thus the exchanger becomes prone to fouling. Fouling is the accumulation of unwanted material on solid surfaces. As the layer of material grows, the thermal conductivity of the surface decreases. Eventually, the fouling layer may hinder the flow of fluid, clogging the heat exchanger and forcing a halt in production. A common problem related to fouling in practice is that the reduced thermal conductivity makes the coolant demand in order to maintain the desired process outlet temperature exceed the coolant supply. Such problems are particularly relevant for seawater coolant systems during spring and summer when coolant inlet temperatures are high.

In general, fouling can be classified as either macro fouling or micro fouling[40]. Macro fouling is caused by biological or inorganic matter like algae, mussels, plants, garbage and more. This type of fouling is mainly found in heat exchangers where the coolant is retrieved using pumps from areas like open sea, rivers or lakes. Meanwhile, micro fouling can come from a variety of sources, both on the process and coolant side of the heat exchanger. Some sources of micro fouling are summarized below[40][42].

- **Precipitation or scaling fouling** occurs when salts, oxides or hydroxides are crystallized on the heat transfer surfaces. Crystallization happens when the solubility limit is exceeded, meaning that the amount of solute is greater than what can be dissolved in the solvent.

- **Particulate or chemical reaction fouling** is caused by particles in the fluid other than salts and hydrates attaching to the heat transfer surface, for example by coagulation.
- **Chemical reaction fouling** happens when substances in the fluid react with the metallic heat exchanger surface, forming a fouling layer.
- **Corrosion fouling** occurs when corrosion is formed on the heat exchanger surface, reducing the conductivity.
- **Solidification fouling** is caused by temperatures on the heat transfer surface reaching the freezing point and thus freezing components of the fluid, such as ice or wax.
- **Biological fouling** happens when microorganisms, shells, plants and other materials accumulate in the heat exchanger. This type of fouling is particularly relevant in open-loop coolant systems, such as when using seawater.

Each type of fouling may happen on either the warm or cold side of the heat exchanger, although for oil and gas processing some types are more common than others. The heat exchangers discussed in this thesis are mostly affected by biological and precipitation fouling in the coolant, and solidification and precipitation fouling in the hydrocarbon fluid.

In addition to decreasing the conductive capabilities of the heat transfer surface, fouling may hinder the flow of fluid by reducing the cross section area of the heat exchanger. If the process flow rate is unaffected by the fouling conditions, reduction in cross section area leads to increased flow velocity, causing increased pressure drop. There are also filters throughout the heat exchanger system in which fouling may accumulate, increasing the pressure drop without affecting the thermal conductivity of the heat exchanger[41].

Fouling in heat exchangers may cause economic losses in the form of reduced production and increased maintenance costs. If the degree of fouling is so severe that the desired process outlet temperature cannot be reached with the maximum available cooling resources, the facility may have to shut down production. If the coolant is distributed from a common source, extensive use of coolant may harm other components of the facility by reducing their available coolant resources. Even at lower levels of fouling, the reduction in heat transfer capability increases the need for coolant resources, which may be financially wasteful.

Heat exchangers typically see gradual decreases in performance over time due to continuous buildup of fouling. More sudden cases of fouling may occur in specific situations, for example if temperature is lowered and large amounts solidification forms. Buildup of fouling may be partially avoided by operating at conditions which avoid liquid or solid formation, and by maintaining sufficient velocity through the heat exchanger for matter not to stick on the heat transfer surface. However, high flow velocities are also undesirable due to increased erosion. These considerations illustrate why optimal heat exchanger sizing is important in practice, to allow maximum flow capacity while avoiding fouling conditions.

3.2.4 Measurements

Sensors are installed to measure properties of the hydrocarbon and coolant fluids. Some important measurements in and around heat exchanger systems used for offshore gas processing are as follows:

- Hydrocarbon fluid inlet/outlet temperature and pressure
- Hydrocarbon fluid pressure drop
- Hydrocarbon fluid flow rate
- Hydrocarbon fluid composition
- Coolant inlet/outlet temperature and pressure
- Coolant pressure drop
- Coolant flow rate
- Coolant valve opening
- Various upstream and downstream measurements

Accurate measurements or estimations of temperature, pressure, flow and composition are vital in ensuring reliable calculations can be performed. In general, all measurement uncertainty will be carried over to any calculations performed using those measurements. Sensor malfunctions may also occur, in which case backup solutions for condition monitoring may be required until the sensor can be repaired and use of the normal monitoring scheme can be continued.

Pressure may be measured through the use of hydrostatic methods like in manometers, aneroid gauges which measure pressure by flexes in metals, or electronic pressure instruments. Temperature cannot be measured directly, meaning a value has to be inferenced from other sources such as a thermometer or electric resistance device. The resistance of certain materials change based on temperature, and thus resistance can be measured using an ohmmeter. An expensive and accurate option is to use platinum resistance thermometers. Mass flow is considered challenging to measure accurately, at least without considerable cost. Common techniques rely on indirect calculations of some kind. Examples are ultrasonic, thermal, turbine, pressure difference or vibration measurements. Composition is rarely known in real-time. Composition measurements can be challenging to acquire, typically relying on techniques such as chemometrics, topological modelling, or infrared analysis.

As mentioned, the hydrocarbon fluid outlet temperature and coolant valve opening are control parameters of the heat exchanger process, and are thus almost always monitored. The flow rate of hydrocarbon fluid is usually known, although with some uncertainty due to anti-surge control and alternative flow channels that bypass the heat exchanger. For heat exchangers with significant limitations on maximum pressure difference, pressure difference sensors are usually installed. Composition is rarely measured in relation to heat exchangers, meaning estimates must be used. Sensors for other parameters may be installed as needed, although a minimal set of sensors is desired to save installation and monitoring costs.

3.2.5 Condition monitoring and maintenance

For maintenance in general, differentiation is made between corrective and preventive maintenance schemes. The former performs repairs after a fault has already been detected, while the latter attempts to avoid fault by performing maintenance ahead of failure. System failure is costly because it halts production and can require expensive repairs. Time spent waiting for and performing the repairs can have economical consequences exceeding those of a preventive repair. Additionally, component failures can cause ripple effects, affecting connected components. Thus, preventive maintenance is generally desired in oil and gas processing.

Sufficient heat exchanger performance is maintained by cleaning the acquired fouling through different methods. High pressure flushing can be used to clear unwanted material in general, while chemical washing with acids is typically performed to remove biological fouling. Some heat exchangers may also be disassembled so that its internals can be more thoroughly cleaned.

To estimate when and how thorough maintenance should be performed, condition monitoring must be implemented. For heat exchangers, condition monitoring means observing the system variables over time and estimating the decrease in performance due to fouling and possibly other factors. This has increased planning- and implementation cost and complexity compared to corrective maintenance techniques, however the benefit of avoiding unexpected system failure often outweigh these concerns. Especially for offshore and potentially unmanned oil and gas facilities, continuous condition monitoring and maintenance planning is essential to avoid unexpected faults. Advancements in measuring technology and digital monitoring tools facilitate the implementation of such schemes. For facilities in hazardous conditions where troublesome operation or maintenance may be expected, simplistic design patterns are often desirable to reduce complexity and thus limit faults[40].

More specifically, preventive maintenance can be partitioned into the following subcategories[43]:

- **Time-based maintenance**: A time interval for inspection is set for the system. This can be effective given an optimal choice of the maintenance interval, e.g. knowing the lifetime distribution of a component. However, if knowledge regarding the system operation does not suffice to find such an optimal interval, inspections can be performed with little benefit. For oil and gas processing specifically, time-based maintenance may be performed at yearly or biyearly inspections.
- **Condition-based maintenance**: If real-time measurements are available, the condition of the system may be addressed continuously. Condition-based maintenance compares measurements to predefined parameter thresholds. This relies on accurate sensors and definitions of the acceptable and unacceptable states, and thus required extensive domain knowledge to optimize.

– **Predictive maintenance**: Although similar to condition-based maintenance, predictive maintenance relies on modelling or formulating precise methods to evaluate the likelihood of errors occurring within a certain time period. For instance, models to estimate remaining useful life (RUL) can be defined to match a desired level of premature repairs (false alarm) or system failures (non-detection). As for condition-based maintenance, extensive domain knowledge may be required to determine the suitable models.

Heat exchangers are historically maintained according to time-based or condition-based maintenance schemes. Measuring the level of fouling directly is impossible, while estimating it accurately has proved troublesome in practice. Hence, the benefits of condition-based monitoring are limited by challenges in defining the required parameter thresholds. Maintenance is often planned according to the experience and interpretations of supervising engineers.

Performance of heat exchangers is typically estimated by calculating thermal effectiveness. For such calculations to be carried out, several assumptions and estimations about the fluid parameters must be made. In practice, this poses multiple challenges, most importantly lack of measurement equipment for the coolant fluid. Additionally, finding accurate expressions for heat transfer and pressure drop when dealing with two-phase flow is extremely difficult. This is part of the reason why such methods have proven unreliable in practice[1]. A common analytical method for calculating heat exchanger performance is presented below. Relevant symbols are explained in Table 3.4.

The Effectiveness-NTU method relates the number of transfer units NTU to the overall heat transfer coefficient U by defining the effectiveness E as the ratio between actual heat transfer q and maximum possible heat transfer q_{max}:

$$E = \frac{q}{q_{max}} = \frac{C_h(T_{h,i} - T_{h,o})}{C_{min}(T_{h,i} - T_{c,i})} = \frac{C_c(T_{c,o} - T_{c,i})}{C_{min}(T_{h,i} - T_{c,i})} \tag{3.18}$$

$$C_h = \dot{m}_h c_{p,h} \tag{3.19}$$

$$C_c = \dot{m}_c c_{p,c} \tag{3.20}$$

$$C_{min} = min(C_h, C_c) \tag{3.21}$$

NTU depends on the flow arrangement in the heat exchanger. For counterflow, meaning the two fluids flow in opposite directions, the following equation holds:

$$NTU = \frac{1}{C_r - 1} \ln \frac{E - 1}{EC_r - 1} \tag{3.22}$$

$$C_r = \frac{C_{min}}{C_{max}} \tag{3.23}$$

NTU is related to the overall heat transfer coefficient by the following equation:

$$NTU = \frac{UA}{C_{min}} \tag{3.24}$$

When used in practice, the expected thermal effectiveness is calculated separately for a clean and dirty heat exchanger. By calculating the actual thermal effectiveness and comparing with the effectiveness for the clean and dirty case, a degree of fouling may be defined. Based on this level and other considerations, maintenance can be planned by supervising engineers.

Table 3.4: Symbols used in heat exchanger performance calculations

Symbol	Definition	Unit
E	Effectiveness	$-$
q	Heat	W
q_{max}	Maximum possible heat transfer	W
T	Absolute temperature	K
C	Heap capacity	$-$
C_r	Heap capacity ratio	$-$
c_p	Specific heat capacity	J/kgK
$\dot{m}$	Mass flow	kg/s
NTU	Number of transfer units	$-$
A	Heat transfer area	m^2
U	Overall heat transfer coefficient	W/m^2K

3.2.6 Predictive maintenance modelling

To derive models used for predictive maintenance, statistical or data-based approaches are often used[43]. The former relies on statistical methods and probability theory, while the latter builds models directly on acquired data.

Statistics are important in different types of maintenance where failure modelling and lifetime estimations are derived using statistical schemes and probability models. Model-based or purely statistical approaches may be suitable if considerable knowledge about the system and its fault model can be obtained. For instance, if failures can be shown to occur at a pattern similar to a Gaussian-, T- or gamma distribution with approximated means and variances, statistical schemes are likely to yield satisfying results. Figure 3.7 shows an example where remaining useful life is estimated by simulating the degradation process. Using the timestamps for which each simulation reaches the maximum degradation level L, a probability density function for the failures can be fitted. Notice that the variance is much smaller for simulations starting from a known degradation level $x(t_j)$ at time t_j,

rather than when starting at time t_0. This type of modelling requires that the chosen simulation process is accurate.

For a number of systems and applications, it can be challenging or even impossible to model the degradation process in the manner described previously. In such cases, data-driven approaches may be used, for which models are build directly from data[43]. Machine learning is one way of building such models. In their fundamentals, machine learning methods are not much different from traditional statistical methods, relying on maximization of a likelihood function to approximate a posterior probability given a provided dataset[44]. They share many areas of use with ordinary statistics, some being classification, regression modelling, and discovering patterns within datasets, but differ in that model fitting and definition is performed according to desired network architectures and parameters, with statistical definitions and calculations hidden from the user.

Ultimately, the use of data-based modelling does not eliminate the need for domain knowledge. Labeling the input data for, and interpreting results from, machine learning models still requires an understanding of the underlying system. In classification systems, the data must manually be labeled to indicate the healthy and unhealthy operation condition before a model can be fitted. For heat exchanger degradation, this is largely infeasible, because defining acceptable and unacceptable behavior is difficult. For regression models, the validity of the output values must be evaluated. Although data-driven approaches like machine learning may be used to find monitoring models, a suitable preventive maintenance scheme using these models must still be defined.

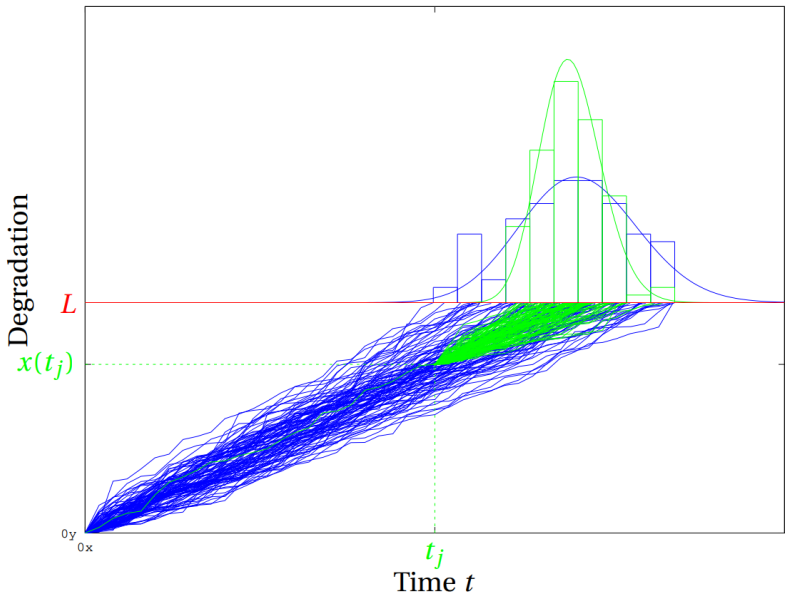


Figure 3.7: Example of density RUL estimation [43, p. 42]

3.3 Machine Learning

In this section, a broad overview of available theory on statistical analysis, machine learning and neural network methods is presented. The content is largely based on a selection of books, papers and video series, namely Murphy [44], Goodfellow *et al.* [45], Haykin [46], Bishop [47], Chollet [48], Hastie *et al.* [49], Brockwell and Davis [50], Bhattacharyya and Kalita [51] and Ng [52]. Individual references are included where deemed appropriate.

When discussing machine learning, the terms method, algorithm, system and model can be easily confused. A method is a procedure for approaching a problem. An algorithm is a specific series of well defined steps for solving a problem. A system includes the algorithm and any underlying frameworks, like the implementation or computational resource. A model is a trained instance of an algorithm.

3.3.1 Introduction

Machine learning algorithms are algorithms capable of learning from data[45]. The goal of machine learning is to develop methods that can automatically detect patterns, and then use these patterns to make predictions on future data or derive other outcomes of interest. A machine learning system is trained rather than explicitly programmed[48]. The training process results in a machine learning model, which can be used to make predictions. Algorithms are designed so that a performance measure, for instance an error metric, reduces when the algorithm gains experience by training on provided data. The resulting model should perform well on previously unobserved inputs, known as its ability to generalize. Machine learning algorithms can be used to solve problems like classification, regression, sequence generation, object detection, speech recognition and image segmentation[48].

Machine learning as a field was developed separately by both AI and statistical researchers[49]. Thus, machine learning is closely related to the fields of statistics and data mining, but differs slightly in terms of its emphasis and terminology[44]. Most importantly, machine learning tends to deal with large, complex datasets for which classical statistical analysis is impractical. Furthermore, machine learning is considered more of a hands-on discipline, relying on empirical proof more often than theoretical[48].

Most machine learning algorithms can be described as using a sample of provided data, a cost function, an optimization procedure and an architecture, to produce a trained model applied to new data samples. When training, the algorithm searches through a set of parameters called the hypothesis space. Algorithms are separated into categories according to their approach, input- and output type, and area of applicability. Machine learning is usually divided into the three main types of supervised learning, unsupervised learning and reinforcement learning.

Supervised learning describes tasks where both the input and desired output for a set of data are supplied to the algorithm. The output may be categorical or nominal. The former indicating a classification problem, while the latter indicates a regression problem. The output may also be a labeled space with natural ordering, although this is more rare. Supervised methods look to learn the function f which maps the input $\boldsymbol{x}$ to the output $\boldsymbol{y}$, based on a labeled set of input-output pairs D:

$$f(\boldsymbol{x} \mid D = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}) = \boldsymbol{y} \tag{3.25}$$

Unsupervised learning does not require a defined set of output parameters. Instead, a set of input parameters are used to discover patterns within the data, known as knowledge discovery. Such patterns may be clusters of data, latent factors, parameter correlation or similar statistical properties of interest. An important application of unsupervised learning is the process of dimensionality reduction, either through feature selection or feature extraction. This is discussed in section 3.3.2. Unsupervised learning can also be used for outlier mining[51].

Supervised and unsupervised learning may be combined, giving rise to semi-supervised learning. For semi-supervised learning problems, a small number of labeled samples are used, in addition to a larger number of unlabeled samples[51]. An example of this may be a set of features with corresponding true or false outputs, for which only samples with an output value of true are present in the dataset. This approach is useful for clustering models, data visualization and more[51].

Reinforcement learning focuses on software agents and their actions in environments. Agents are not equipped with the exact output of the learning problem, but rather an utility function which rates a state in the environment based on certain metrics. Based on its utility function, an agent can choose actions in order to increase its utility and thus achieve desirable performance in its environment. A feedback loop between the environment and the learning system must be implemented. Agents should be autonomous, meaning they must learn to perform tasks by trial and error, without the guidance of human operators. Examples of problems where reinforcement learning has been implemented with great success are for games like chess and go[53].

This thesis focuses primarily on unsupervised learning in the form of dimensionality reduction and data visualization, and supervised learning in the form of regression and neural networks. Dimensionality reduction is discussed in section 3.3.2, regression in section 3.3.3, and neural networks in 3.3.4 and onward.

Machine learning systems contain internal and external parameters. External parameters, also called hyperparameters, affect how the system operates and performs its training. These parameters are set prior to the learning process. Hyperparameters vary greatly for different machine learning algorithms. Examples of some

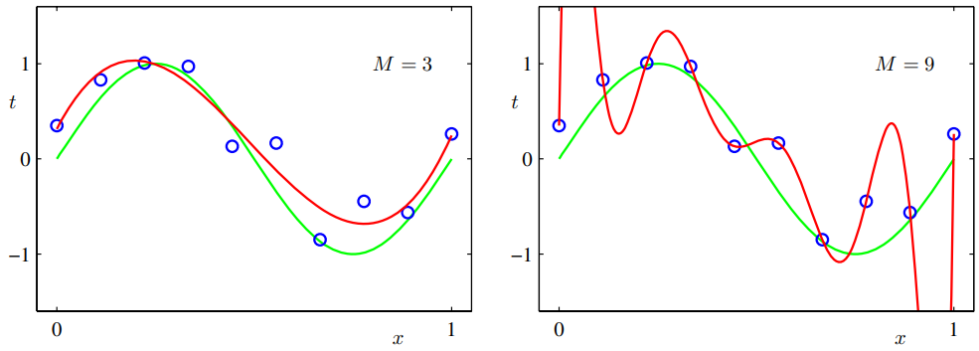


Figure 3.8: Two polynomials of degree $M = 3$ and $M = 9$ fitted to a set of 10 points. The polynomials are plotted in red, while the points are plotted as blue circles. The green curve is a sine function from which the data points were generated together with some added noise [47, p. 7]

parameters are network layers and neurons in neural networks, regularization, activation functions, optimizer, callbacks, learning rates, number of iterations, batch sizes, metrics and more. Some remain constant throughout the training period, such as the model architecture, while others can be adjusted, like the learning rate. Choosing optimal hyperparameters can be very challenging in practice. This problem is discussed further in section 3.3.12. Internal parameters are values adjusted during model training, and contribute in calculating the output of the model. Examples of such parameters are the weights of a neural network or coefficients of a linear regression. Internal parameters are not set by the user, and typically initialized based on some predefined scheme. The effect and usage of each relevant parameter is explained throughout this chapter.

When fitting a machine learning algorithm, a very complex function may be produced that performs well on the data the model was trained on, but fails to generalize for data samples not in the training set and thus worsen the performance on validation or testing data. This is known as overfitting, indicating that the model has failed to generalize and is only capable of accurately modelling the data it was trained on. An example is seen in Figure 3.8, where polynomials of degree 3 and 9 are fitted to a set of 10 points. The corresponding training and test metrics, here the root mean squared error, can be seen in Figure 3.9. The ideal choice of polynomial degree for this example is in the range 3 to 7, for which both training and testing error is low.

Overfitting of models must be prevented, and is discussed further in sections 3.3.3 and 3.3.4. Unsupervised learning is typically less prone to overfitting than supervised learning[54]. The opposite of overfitting is called underfitting, and indicates that the model has not yet learned a satisfying representation of the training data. In such cases, the capacity and thus learning capabilities of the model should be increased[48].

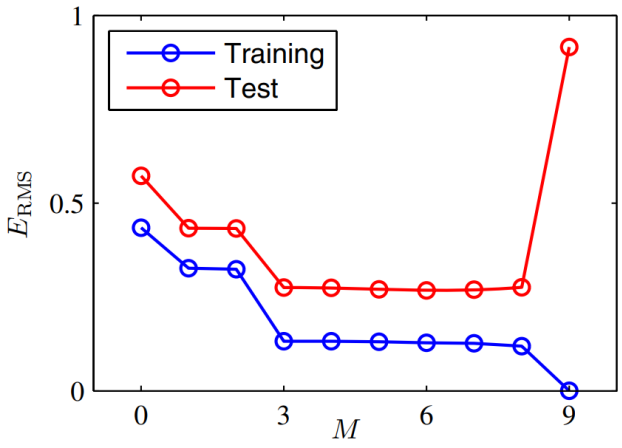


Figure 3.9: Training and test error for polynomials of degree M for the points shown in Figure 3.8. The test error increases drastically as the training error approaches zero, indicating that the model has overfitted to the provided training data [47, p. 8]

3.3.2 Dimensionality reduction

Dimensionality reduction can be used to find a representation of data that has lower dimensionality than the original input. This is often desirable in order to reduce computational complexity, eliminate information redundancy, increase accuracy, facilitate data understanding and improve generalization[51]. Machine learning problems become exceedingly difficult when the number of dimensions in the data increase due to the curse of dimensionality[55]. When the number of features increase, the number of distinct configurations increase exponentially. Thus, it becomes increasingly difficult for a model to cover the entirety of the feature space. Feature selection or feature extraction may be used to reduce the dimensionality.

Feature selection is the process of reducing the dimensionality by simply discarding features according to some appropriate scheme. In doing so, a reduced selection of the original data features is used to represent the data. Ideally, this selection should capture as much intrinsic information from the original data as possible. How to discard features varies for each dataset and domain, and often requires expert knowledge[49]. A general approach is to remove features that are either entirely correlated with other features, or have almost no variation at all. Entirely correlated features are likely to be governed by the same underlying physics, and thus monitoring of one such feature is often sufficient. Features with almost no variation are unlikely to affect other parameters, essentially making them constants and therefore excess parameters.

Feature extraction is the process of extracting new features that do not necessarily align with the original features. This can be an effective technique for data

representation and visualization[46]. A popular technique is the use of Principal Component Analysis, which builds a set of orthogonal linear basis vectors. This set is chosen so that it minimizes the reconstruction error when a subset of features are used to reconstruct the original data. The optimal solution is obtained by choosing the eigenvectors of the feature covariance matrix with the largest eigenvalues. Given an m-dimensional data vector $\boldsymbol{x}$, the data can be represented by an l-dimensional vector of principal components $\boldsymbol{a}$ as follows:

$$\boldsymbol{a} = \boldsymbol{q}^T \boldsymbol{x} \tag{3.26}$$

where the ith element of vector $\boldsymbol{q}$ is the ith eigenvector of the correlation matrix $R = E[\boldsymbol{x}\boldsymbol{x}^T]$, and the ith element of vector $\boldsymbol{a}$ is the projection of the data vector $\boldsymbol{x}$ onto the ith eigenvector q_i. If $l = m$, then the new vector $\boldsymbol{a}$ is a rotated version of the original data vector $\boldsymbol{x}$. If $l < m$, then only a subset of the eigenvectors are retained, making the representation of data approximate[46]. The amount of the original variance captured in the principal components is referred to as the explained variance ratio of the model.

The data may be reconstructed as follows:

$$\hat{\boldsymbol{x}} = \sum_{i=1}^{l} a_i \boldsymbol{q}_i + \epsilon \tag{3.27}$$

where $\hat{x}$ is the reconstructed vector and ϵ is the reconstruction error. ϵ is equal to zero if $l = m$. The accuracy of the reconstruction is similar to the explain variance ratio of the PCA decomposition.

An example of PCA can be seen in Figure 3.10. In this example, no dimensionality reduction is performed, but rather a change of coordinate system. Even so, the same principal applies for data of higher dimensions. Axis corresponding to the most significant eigenvectors are chosen. For data of higher dimensionality, reducing data in this manner to two or three dimensions enables plotting, which can be of great use for data visualization. Data visualization in itself can be used as a form of anomaly detection, by inspecting the location of new points in relation to historical data.

A disadvantage of feature extraction is that meaningful practical and domain specific information like feature units and sensor placement is lost when abandoning the original data features. Thus, results obtained by models using feature selection may be easier to explain than if feature extraction is used. The use of data reconstruction may help reestablish some of this information, however doing so introduces yet another step in the modelling pipeline. In general, feature selection is more applicable for data from physical systems[49].

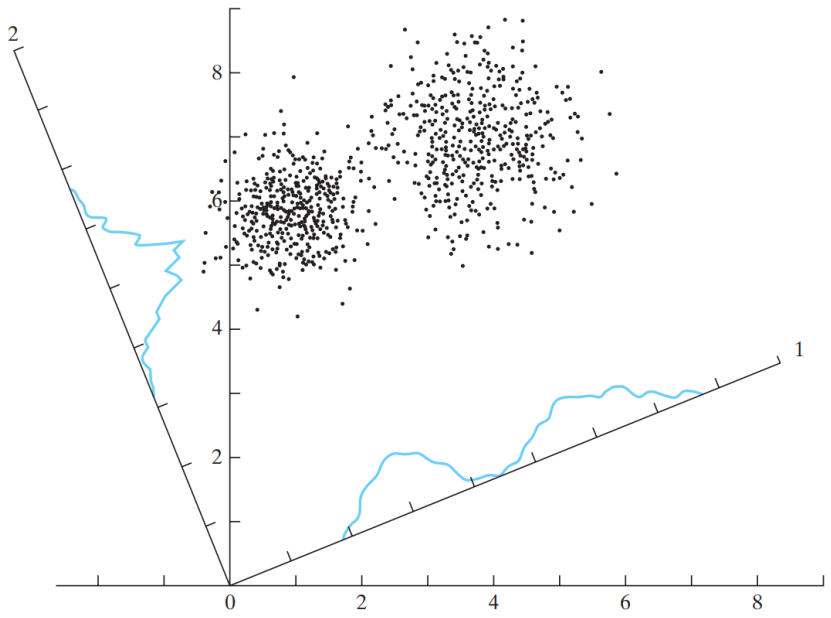


Figure 3.10: In the original coordinate frame, the data samples are distributed in a linearly increasing pattern. When using the coordinate frame of the principal components, two separate groups can easily be distinguished on the x-axis, while the y-axis appears to have a Gaussian distribution [46, p. 381]

3.3.3 Regression models

Given a set of input and output measurements, a prediction model can be built to predict the outcome for unseen objects. The simplest form of such a model is simple linear regression. A weight vector $\boldsymbol{w}$ is determined, which maps the input vector $\boldsymbol{x}$ consisting of D features to the output $\boldsymbol{y}$ given some residual error ϵ. This may be written as follows for the case where $\boldsymbol{y}$ is a single decimal value:

$$y(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{x} + \epsilon = \sum_{j=1}^{D} w_j x_j + \epsilon \tag{3.28}$$

Functional transformations for the input features may be introduced in the linear regression expression through the use of basis functions ϕ. The primary reason for using such functions is that the input data may be transformed to a function space in which it is linear in the transformed variables, enabling the use of linear models[49]. The introduction of basis functions is called linear basis expansion:

$$y(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}) + \epsilon = \sum_{j=1}^{D} w_j \phi_j(x_j) + \epsilon \tag{3.29}$$

ϕ_j is the basis function for element j. Such basis functions may be polynomial, logarithmic, exponential, piecewise functions and more. Thus, a large area of problems may be solved using linear basis expansion and linear methods. Which basis functions to apply must be adapted for each task and domain.

The weight vector $\boldsymbol{w}$ is typically found through minimization of some error function, also commonly referred to as a loss function. Finding the set of weights which minimizes the error increases the model accuracy. For regression problems, the error function often uses the difference between desired and calculated output, $y_i - \boldsymbol{w}^T \boldsymbol{x}_i$. Errors functions containing the square of this difference are most commonly used in practice[49], such as in the residual sum of squares:

$$RSS(\boldsymbol{w}) = \sum_{i=0}^{N} (y_i - \boldsymbol{w}^T \boldsymbol{x}_i)^2 \tag{3.30}$$

The residual sum of squares may also be divided by the number of samples to obtain the mean squared error, which is more comparable between datasets of different sizes. This is the error function most commonly used for regression tasks[48]:

$$MSE(\boldsymbol{w}) = \frac{1}{N} \sum_{i=0}^{N} (y_i - \boldsymbol{w}^T \boldsymbol{x}_i)^2 \tag{3.31}$$

Furthermore, the root may be used to derive the root mean squared error:

$$RMSE(\boldsymbol{w}) = \sqrt{\frac{1}{N} \sum_{i=0}^{N} (y_i - \boldsymbol{w}^T \boldsymbol{x}_i)^2} \tag{3.32}$$

The root mean squared error is sometimes preferred because it ensures the error has the same unit as the original output feature[47]. If more than one output is used, these are likely to either have different units or different value ranges, and so the use of root mean squared error is not as useful or important.

The residual error for regression models is often assumed to have a Gaussian distribution with zero mean and constant variance, meaning $\epsilon \approx \mathcal{N}(\mu, \sigma^2) = \mathcal{N}(0, k^2)$ for some k. When this is the case, minimizing the sum of squared residuals finds the optimal solution. However, datasets often contain outliers that cannot be trivially removed, as well as a limited number of samples. This means the assumptions made for the Gaussian distribution do not necessarily hold. In such cases, the use of a sum of squared residuals scheme may result in a poor fit[49]. Robustness to outliers can be achieved by replacing the Gaussian distribution with a different distribution. This makes the optimization process of the regression model much less trivial. Alternatively, an absolute rather than squared error metric may be used, such the mean absolute error:

$$MAE(\boldsymbol{w}) = \frac{1}{N}\sum_{i=0}^{N}|y_i - \boldsymbol{w}^T\boldsymbol{x}_i| \tag{3.33}$$

Particularity in deep learning problems, which will be discussed shortly in section 3.3.4, samples sizes are often very large, perhaps hundreds of thousands or millions of dataset rows. When the number of random variables from the same population becomes large, the distribution will tend towards a Gaussian distribution according to the Central Limit Theorem[47]. This suggests the use of squared metrics can be justified also in the presence of outliers. Even so, it is important to be aware that not all outliers distributions seen in practice are random or have zero mean. For example, a flow measurement sensor will be bias towards measuring a flow value of zero in the absence of a measurement signal. Hence, numerous outlier readings may indicate a flow value of zero. If all other outliers in the same set are zero-mean, then the overall mean of outliers in the dataset will be negative due to the extreme outliers caused when signal is absent. The handling of outliers in general is extremely difficult. Expert knowledge of the underlying process is often required to determine appropriate handling.

The global minimum of the loss function is typically always an overfit solution, meaning the modelled function represents the training data very well, but without the ability to generalize for samples not used for training[49]. Hence, rather than finding the parameters which optimize the loss function, the generalizing capabilities of the model must be optimized. The overall error E of a machine learning model will thus be of the form $E = bias + variance + e_\infty$, where e_∞ is an irreducible error due to data noise. Bias, also called approximation error, indicates the ability of a model to fit the training data, and is thus incurred in the training of the model itself. Variance, also called estimation error, indicates how well the model generalizes to new samples not used for training. Hence, models must be optimized according to a tradeoff between bias and variance, called the bias-variance dilemma[46]. This states that both bias and variance may only be eliminated if the number of training samples becomes infinitely large.

To improve generalization, regularization is introduced. In traditional regression, the most common approach is to include a penalty term for large weights in the optimization function, known as weight regularization. With this penalty term, the error of a regression model takes the following general form:

$$E(\boldsymbol{w}) = E_D(\boldsymbol{w}) + \lambda E_W(\boldsymbol{w}) \tag{3.34}$$

E is the total error, E_D is the error calculated on the predicted and measured output, and E_W is the error term introduced by regularization. Using least squares error, basis function $\boldsymbol{\phi}$ and a polynomial regularization error term, the general

error function becomes the following:

$$E(\boldsymbol{w}) = \sum_{i=1}^{N} (y_i - \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}))^2 + \frac{\lambda}{2} \sum_{j=1}^{M} |w_j|^q \tag{3.35}$$

N is the number of observations, M is the number of weight parameters, and q is the regularizing term. $q = 1$ is known as LASSO regularization, while $q = 2$ is known as ridge regularization. Finding an optimal value for the regularization coefficient λ is very difficult[47]. A large coefficient will make the regularization term dominate the predicted error term, and thus make learning impossible as any adjustments made to the weights results in increased total error. Trial and error is often necessary to find a suitable regularization coefficient.

3.3.4 Artificial neural networks

Artificial neural networks, commonly referred to as just neural networks, uses the same idea of fitting weights as in the case of traditional regression models. However, neural networks do so at a much larger scale and in multiple layers. These networks are vaguely inspired by the biological neural networks in the brain, although there is no evidence that the brain implements anything like the learning mechanisms used in modern machine learning models[48]. Some useful properties and capabilities of neural networks are nonlinearities, input-output mapping, adaptivity and fault tolerance[46], as well as high degree of connectivity[46]. However, these same properties make theoretical analysis and interpretability of multilayer networks difficult.

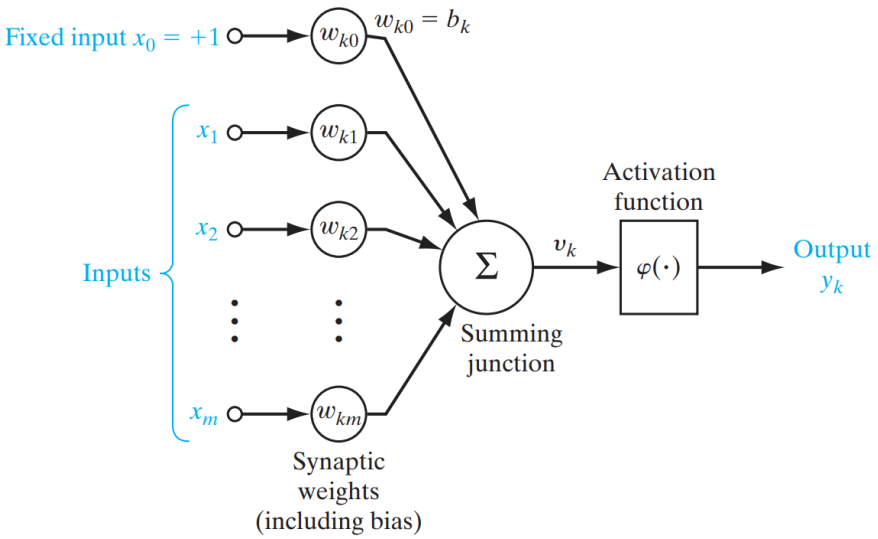


Figure 3.11: Unit or neuron used in artificial neural networks [46, p. 12]

The fundamental building blocks of neural networks are units or neurons. These two terms are used interchangeably. Each neuron has a structure similar to what

is illustrated in Figure 3.11. A set of inputs x_i for i in the range 1 to m, as well as a fixed bias parameter x_0, are multiplied by a corresponding weight and summed at a summing junction. An activation function ρ is applied to the sum, whose output is the final output y of the neuron.

The bias parameter is added so that the output is biased towards being equal to the bias in the absence of any input. This is essential in ensuring successful learning[46]. The effect of the bias can be thought of equivalently as the constant term k in a function $y = f(x) + k$, shifting the output value y along the x-axis.

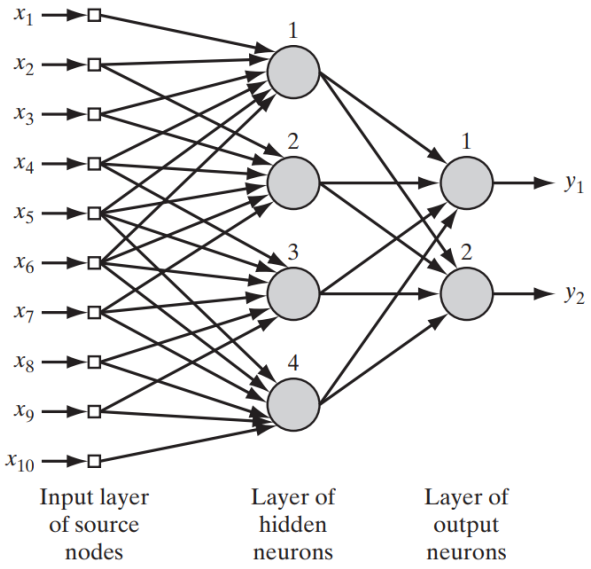
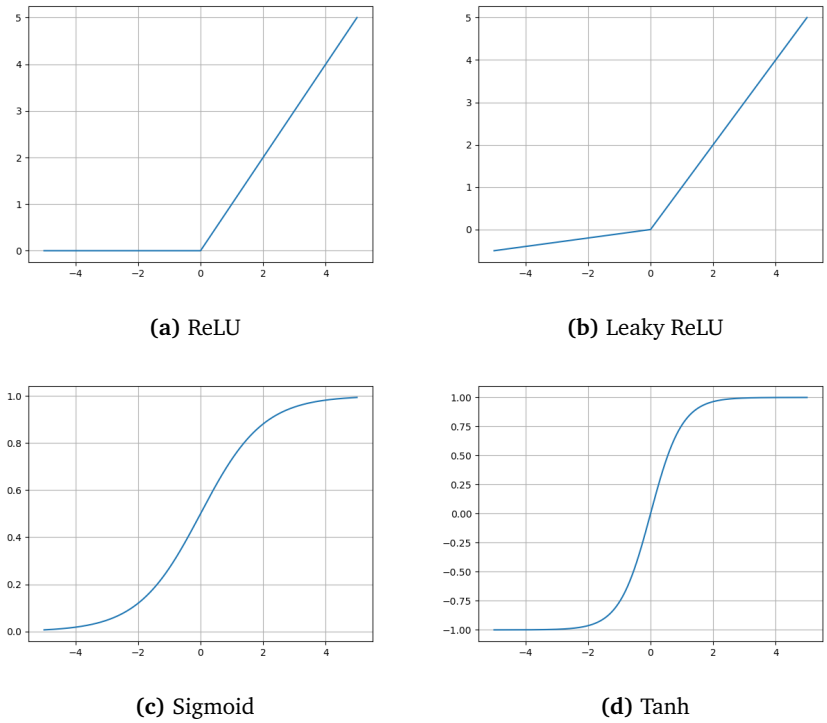


Figure 3.12: Artificial neural network with input features, one hidden layer of four units, and an output layer of two units [46, p. 29]

In a neural network, several neurons are connected in a series of layers as seen in Figure 3.12. Units in the input layer are plain input values, while units in the remaining layers are neurons with the structure previously described. Dense layer connections are commonly used, also referred to as a fully connected network. In such networks, each unit is connected to every unit in the following layer. The layers between the input and output layers are referred to as hidden layers. There may be any number of hidden layers. The total number of layers is called the network depth, while the number of neurons in each layer is called the layer width. Separate layers may have a different widths.

The activation function at each neuron is chosen based on desirable features. Typically, one activation function is used for every hidden neuron, while another activation function is used for the output neurons. Some commonly used activation functions are seen in Figure 3.13. Nonlinear activation functions are required in the hidden neurons for networks to learn nonlinear transformations. Nonlinear transformations give a richer hypothesis space, enabling deeper represent-



(a) ReLU **(b)** Leaky ReLU

(c) Sigmoid **(d)** Tanh

Figure 3.13: Activation functions commonly used in neural networks.

ations[48]. Other desirable properties are that the functions should saturate in neither region, increasing the expressibility of each neuron. Leaky ReLU saturates in neither region, while ReLU saturates in the negative region. Sigmoid and tanh saturate in both regions. ReLU is the default activation function recommended for use with most feedforward neural networks[45].

The universal approximation theorem[56] states that any neural network with at least one hidden layer of nonlinear activation functions can represent any continuous function, making such neural networks universal approximators. However, doing so for complex functions using only one hidden layer may require an exponential number of neurons. Thus, deeper architectures are often desirable[45]. Note that the theorem states nothing about the learnability of the representation. Learning this representation, even using deep architectures, may not be trivial.

The activation function applied to the output neurons of a neural network is chosen based on the type of predictions the network should perform. For binary classification, an output value between 0 and 1 is required. Thus, the sigmoid function may be used. For classification with several output classes, softmax is typically used, seen in Eq. 3.36. Softmax calculates a probability distribution over the possible output classes, where K is the number of classes and e is the exponential function. For regression tasks, a linear activation function must be applied.

$$\rho(\boldsymbol{x})_j = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}} \tag{3.36}$$

For a set of input values, the output of the neural network is calculated in a feed-forward manner. Weights are fixed while the values applied to the input units are propagated through the network, calculating the sum and applying the activation function at each neuron. The final output is calculated in the output layer.

For the neural network to learn and improve its performance according to the chosen error metric, weights must be adjusted in a way which reduces the error. Weights are updated according to an optimization algorithm called an optimizer, which typically includes calculating and using the gradient of the error function. Optimizers and the process of calculating the gradient is discussed in section 3.3.6. Prior to calculating the gradient, a way of propagating changes in the error function backwards in the neural network is required, discussed in section 3.3.5.

Overfitting, as discussed for regression models in section 3.3.3, is an essential concern in neural networks as well. The same weight regularization techniques as discussed previously can be used, such as LASSO and ridge regularization. Additionally, there are some specific regularization techniques only applicable for neural networks. Dropout has proven to be one of the most effective and most commonly used such techniques[48]. When using dropout, units in the neural network are switched off for one iteration at random throughout the training process. This means nearby neurons will not be able to co-adapt, and thus increase regularization by restricting dominating network relations. In practice, this means training and combining a large subset of different neural networks, although in a much more computationally and memory efficient way than what would have been possible when training separate networks. Typically, about $20-50\%$ of the units in each layer are switched off for each iteration[48].

3.3.5 Backpropagation

The backpropagation algorithm allows the derivatives of the error function to be calculated according to each weight. This technique is essential in efficiently training neural networks with many parameters, because calculating gradients and updating weights for one single neuron at the time is horribly inefficient[48]. Errors are propagated backwards according to the following algorithm[47]:

1. Apply an input vector $\boldsymbol{x_n}$ to the network and forward propagate through the network to find the activation of all hidden and output units
2. Evaluate the error δ_k, also called the local gradient, for all the output units
3. Backpropagate δ_k to obtain δ_j for each hidden unit
4. Evaluate the required derivatives and update the model parameters

Given an input vector $\boldsymbol{x}$, the sum of incoming edges for each neuron is calculated as

$$a_j = \sum_i w_{ji} z_i \tag{3.37}$$

where a_j is the sum calculated at neuron j. z_i is the output at the previous neuron i, or $\boldsymbol{x}_i$ when j corresponds to the first layer of hidden units. w_{ji} is the weight of the edge between the neurons. The output of neuron j is calculated by

$$z_j = \rho(a_j) \tag{3.38}$$

where z_j is the output of the neuron and ρ is the activation function. This calculation is performed in a feedforward manner for each neuron in each layer, until the final outputs are calculated. The process up until now corresponds to calculating the feedforward neural network output as described in the previous section.

During the backpropagation algorithm, the error of the outputs δ_k are calculated by

$$\delta_k = y_k - t_k \tag{3.39}$$

where t_k is the target output value and y_k is the predicted value for output parameter k. Errors are propagated backwards by calculating

$$\delta_j = \rho^{'}(a_j) \sum_k w_{kj} \delta_k \tag{3.40}$$

where δ_j is the error of the backward neuron j, $\rho^{'}(a_j)$ is the derivative of the activation function evaluated at neuron j, w_{kj} is the weight between neuron k and j, and δ_k is the error at the forward neuron k. An overview of the backpropagation process for a single backwards neuron can be seen in Figure 3.14.

The partial derivatives of the error function can then be calculated as

$$\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i \tag{3.41}$$

which states that the change in error E_n in relation to the weight w_{ji} is expressed by the backpropagated change in previous neurons δ_j for neuron j and the output z_i of neuron i.

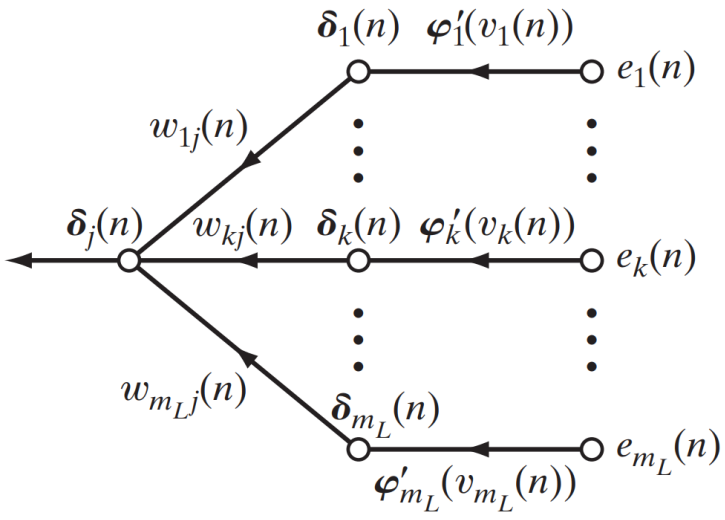


Figure 3.14: Backpropagation for a single neuron j two layers back from the output layer. The error δ_j at node n is calculated according to Eq. 3.40 [46, p. 134]

3.3.6 Gradient descent

For traditional models such as the simple linear regression with least square optimization, closed form solutions for the weight parameters exist. For instance, linear regression using least square methods can be solved as

$$\boldsymbol{w} = (\boldsymbol{x}^T\boldsymbol{x})^{-1}\boldsymbol{x}^T\boldsymbol{y} \tag{3.42}$$

where X^T is the transpose of the input vector. If the inverse does not exist or is difficult to find, the pseudoinverse[57] may be used. However, when nonlinearities are introduced, there are no such closed form solutions. In the nonlinear case, iterative and numerical optimization procedures must be applied to search the parameter space for incrementally better solutions. An iterative procedure updates the weight vector w according to the following general scheme:

$$\boldsymbol{w}_{new} = \boldsymbol{w}_{old} + \Delta\boldsymbol{w} \tag{3.43}$$

The change in weights $\Delta\boldsymbol{w}$ is typically a function of the gradient of the error function. For datasets with thousands or millions of samples, it is impossible to calculate the exact gradient. Therefore, by choosing a subset of sample and calculating the gradient based on these samples, an estimate of the total gradient can be calculated. Doing so is known as batch learning. Larger batches provide more accurate estimates, however at the cost of increased hardware requirements. Batch training also enables parallel computation.

During training, the process of changing the model parameters is controlled by the model optimizer. The optimizer specifies the exact way in which the gradient of the loss will be used to update parameters[48]. A simple optimizer is the Stochastic gradient descent (SGD) optimizer, which uses standard mini-batch SGD. The weights are updated according to the following equation:

$$\boldsymbol{w}_{new} = \boldsymbol{w}_{old} - \eta \nabla L(\boldsymbol{w}_{old}) \tag{3.44}$$

$\boldsymbol{w}$ is the weight vector and η is the learning rate. L is the loss function, calculated using a mini-batch consisting of n samples. n is equal to the batch size specified for the neural network model.

The learning rate η is a positive scalar which determines the adjustments made to the model weights in each iteration. This parameter is particularly important because it can have a significant impact on model training and thus model performance. The learning rate may be changed during training, for example according to a predefined schema. Increasing the learning rate improves the convergence speed of the algorithm, but causes larger fluctuations in the convergence pattern.

Momentum may be used for improved method stability in variations of the basic gradient descent algorithm. Momentum uses a sequence of gradients to average the gradient over a set of iterations, enhancing the training capabilities when the original gradient is either noisy or extremely small. Gradient descent with momentum for one past gradient is as follows:

$$\boldsymbol{w}_{new} = \boldsymbol{w}_{old} - \eta \nabla L(\boldsymbol{w}_{old}) + \alpha(\Delta \boldsymbol{w})_{old} \tag{3.45}$$

Here, $(\Delta \boldsymbol{w})_{old}$ is the update in weights from the last iteration. α is an exponential decay factor that determines the contribution from this past term. By adding this term, the weight vector can be updated according to a linear combination of the current estimated gradient and the previous weight update. This can be expanded further to include additional past weight update terms.

Some popular improvements to the standard gradient descent algorithm are the Adaptive Gradient algorithm (AdaGrad), Root Mean Squared Propagation (RMSProp), and Adaptive Moment Estimation (Adam).

AdaGrad[58] modifies SGD to use an individual learning rate for each parameter. The parameter η still specifies a global learning rate, however a scaling factor for the learning rate corresponding to each weight is found through calculating the diagonal of the outer product matrix G:

$$G = \sum_{\tau=1}^{t} \nabla L_{\tau}(\boldsymbol{w}_{old}) \nabla L_{\tau}(\boldsymbol{w}_{old})^{T} \tag{3.46}$$

G is the outer product of all previous subgradients up until time t, and is updated for each iteration. The update formula for the weight vector is as follows. The symbol $\circ$ is the element-wise product operator, called the Hadamard product. Note that squaring and square-rooting operations are done elementwise in the following equations:

$$\boldsymbol{w}_{new} = \boldsymbol{w}_{old} - \eta \, diag(G)^{-1/2} \circ \nabla L(\boldsymbol{w}_{old}) \tag{3.47}$$

A challenge with AdaGrad is that the diagonal terms of the outer product calculated at each iteration are squared terms and therefore positive. As the number of iterations grow, the sum of these squares increase. The learning rate must become very small to compensate. Eventually, the learning rate may become so small that further learning is impossible.

RMSProp[59], like AdaGrad, adapts learning rates for each parameter. However, instead of accounting for all previous subgradients, a running average is used. β is the forgetting factor or moving average parameter, often chosen to be 0.9. A function $v(x,t)$ is introduced to represent the running average at time step t.

$$v(\boldsymbol{w},t) = \beta v(\boldsymbol{w},t-1) + (1-\beta)(\nabla L(\boldsymbol{w}_{old}))^2 \tag{3.48}$$

$$\boldsymbol{w}_{new} = \boldsymbol{w}_{old} - \frac{\eta}{v(\boldsymbol{w},t)} \nabla L(\boldsymbol{w}_{old}) \tag{3.49}$$

Adam[60] expands further on the idea of using running averages, by introducing a second running average term $m(w,t)$ for the non-squared gradient:

$$m(\boldsymbol{w},t) = \beta_1 m(\boldsymbol{w},t-1) + (1-\beta_1)\nabla L(\boldsymbol{w}) \tag{3.50}$$

$$v(\boldsymbol{w},t) = \beta_2 v(\boldsymbol{w},t-1) + (1-\beta_2)(\nabla L(\boldsymbol{w}))^2 \tag{3.51}$$

$$\boldsymbol{w}_{new} = \boldsymbol{w}_{old} - \eta \frac{m(\boldsymbol{w}_{old},t)(1-\beta_1)^{-1}}{\sqrt{v(\boldsymbol{w}_{old},t)(1-\beta_2)^{-1}} + \epsilon} \tag{3.52}$$

Many implementations of deep neural networks have proven to perform poorly in practice, because the gradient becomes very weak when propagated through the model layers further away from the output[44]. This is known as the vanishing gradient problem. Additionally, there may be large plateaus in the error surface that are hard to avoid while the gradient is decreasing. If the algorithm converges to such a plateau, the performance may not be optimal.

The opposite of the vanishing gradient problem, namely the exploding gradient problem, may also be encountered. In such cases, gradient clipping may be applied in order to avoid abnormally large changes in network weights. By applying gradient clipping, the maximum value of the gradient is limited to a specified level unlikely to cause issues when training.

In general, setting the learning rate prior to training and adjusting it during training is very difficult[45]. A somewhat advanced strategy to try counter the reduced gradient in early layers is to adjust the learning rate for each neuron so that it is inversely proportional to the square root of total connections made to that neuron[46].

3.3.7 Recurrent neural networks

Traditional neural networks have no memory, meaning pairs of input and output are processed independently of each other. For recurrent neural networks, memory is introduced by the addition of at least one feedback loop. This enables sequence modelling, which can be essential for problems like time series modelling or speech and text processing[52]. In sequence modelling, the goal is to predict a future value based on a sequence of past values.

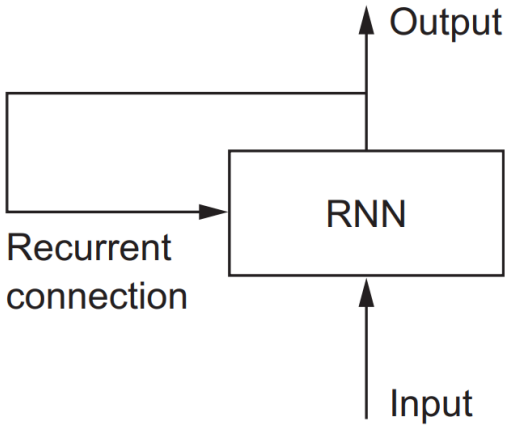


Figure 3.15: Neural networks which contain at least one feedback loop are considered recurrent neural networks. The recurrent connection allows calculations to be performed based on previous states [48, p. 196]

In recurrent networks, a state is maintained between each sample in a sequence of values. This makes each unit capable of modelling long-term dependencies[45]. The state is reset between independent sequences. The general architecture for recurrent neural networks is seen in Figure 3.15. Apart from the calculations performed in the network units, this is largely the same as for the neural networks discussed in section 3.3.4. In recurrent neural networks, a modified version version of dropout called recurrent dropout is used for the recurrent connections. Traditional dropout is still used for the normal inputs.

Recurrent neural networks with simple units like the ones discussed in section 3.3.4, with the addition of a single feedback loop, do not perform well in practice[48]. This is because the vanishing gradient problem becomes even more challenging in recurrent networks, where gradients are propagated over many time steps. Gradients representing long-term dependencies tend to either vanish or explode. Another challenge is that changes in the internal state may not be measurable by the gradient[46]. To manage the vanishing or exploding gradient problem, networks containing more complex recurrent units such as Long Short-Term Memory units and Gated Recurrent Unit (GRU) are implemented.

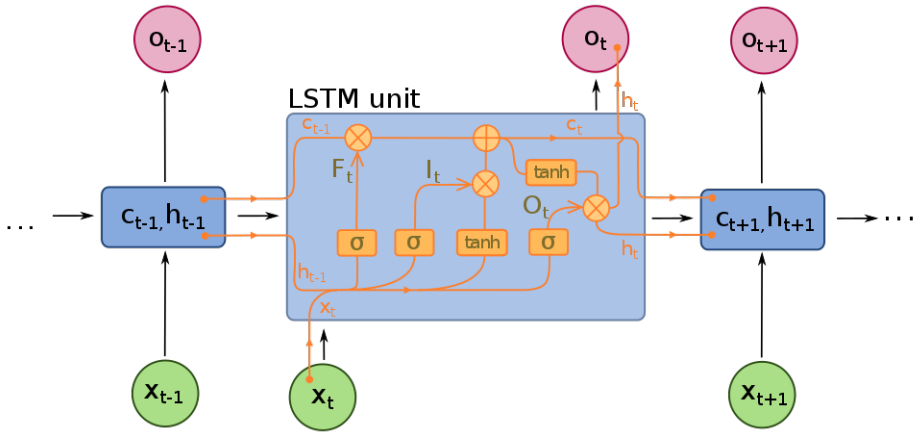


Figure 3.16: A LSTM unit at time step t, between the previous and following time step $t-1$ and $t+1$. The green nodes represent the input, while the red nodes represent the output. The pathway of the carry parameter C and state parameter h can be seen between successive time step, allowing recurrent connections [61]

LSTM units, like the one seen in Figure 3.16, implement a memory or carry parameter $\boldsymbol{c}_t$ and a state parameter $\boldsymbol{h}_t$ in addition to the input parameter $\boldsymbol{x}_t$ and bias parameter $\boldsymbol{b}$. Using a series of activation functions, the different signals are passed through three gates known as input gate, forget gate and output gate. Separate weight matrices $\boldsymbol{W}_q$, $\boldsymbol{U}_q$ and $\boldsymbol{b}_q$ are kept for the current and recurrent connections as well as the bias for each parameter q. The vectors $\boldsymbol{F}_t$, $\boldsymbol{I}_t$, $\boldsymbol{O}_t$, $\boldsymbol{C}_t$ and $\boldsymbol{h}_t$ as indicated on the figure are calculated as follows:

$$\boldsymbol{F_t} = \sigma(\boldsymbol{W}_f \boldsymbol{x}_t + \boldsymbol{U}_f \boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \tag{3.53}$$

$$\boldsymbol{I_t} = \sigma(\boldsymbol{W}_i \boldsymbol{x}_t + \boldsymbol{U}_i \boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \tag{3.54}$$

$$\boldsymbol{O_t} = \sigma(\boldsymbol{W}_o \boldsymbol{x}_t + \boldsymbol{U}_o \boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \tag{3.55}$$

$$\boldsymbol{C_t} = \boldsymbol{F}_t \circ \boldsymbol{C}_{t-1} + \boldsymbol{I}_t \circ \sigma_h(\boldsymbol{W}_c \boldsymbol{x}_t + \boldsymbol{U}_c \boldsymbol{h}_{t-1} + \boldsymbol{b}_c) \tag{3.56}$$

$$\boldsymbol{h_t} = \boldsymbol{O}_t \circ \sigma_h(\boldsymbol{C}_t) \tag{3.57}$$

The initial parameters $\boldsymbol{C}_0$ and $\boldsymbol{h}_0$ are equal to zero. σ is the chosen activation function of the respective gate, traditionally the sigmoid function. σ_h is the hyperbolic

tangent function. It is apparent from these equations that recurrent units calculate and store a large number of weights, increasing the time and space complexity of recurrent algorithms compared to traditional neural network models.

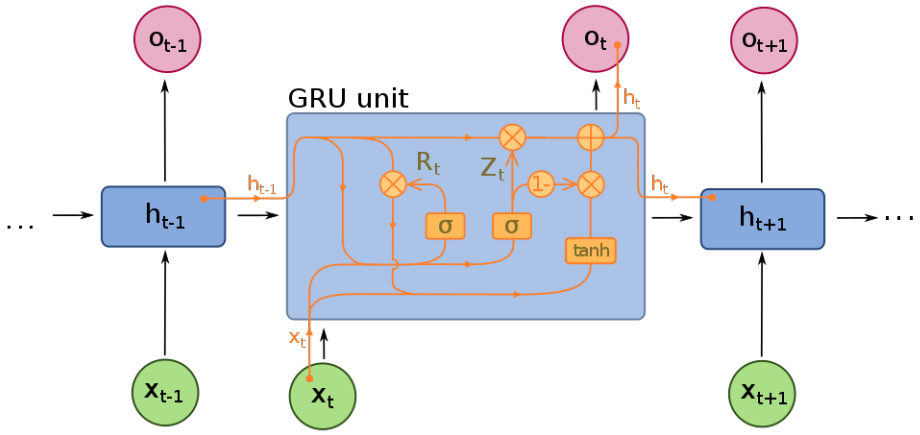


Figure 3.17: A GRU unit at time step t, between the previous and following time step $t-1$ and $t+1$. The green nodes represent the input, while the red nodes represent the output. The pathway of the state parameter h can be seen between successive time step, allowing recurrent connections [62]

GRU uses the same principles as LSTM, although is somewhat more streamlined and thus cheaper to run, at the cost of reduced representational power compared to LSTM[48]. The carry parameter from the LSTM unit is removed, in addition to some changes in the internal computation, as seen in Figure 3.17. The vectors $\boldsymbol{R}_t$, $\boldsymbol{Z}_t$ and $\boldsymbol{h}_t$ as indicated on the figure are calculated as:

$$\boldsymbol{R_t} = \sigma(\boldsymbol{W}_r \boldsymbol{x}_t + \boldsymbol{U}_r \boldsymbol{h}_{t-1} + \boldsymbol{b}_r) \tag{3.58}$$

$$\boldsymbol{Z_t} = \sigma(\boldsymbol{W}_z \boldsymbol{x}_t + \boldsymbol{U}_z \boldsymbol{h}_{t-1} + \boldsymbol{b}_z) \tag{3.59}$$

$$\boldsymbol{h_t} = \boldsymbol{Z}_t \circ \boldsymbol{h}_{t-1} + (1 - \boldsymbol{Z}_t) \circ \sigma_h(\boldsymbol{W}_h \boldsymbol{x}_t + \boldsymbol{U}_h(\boldsymbol{R}_t \circ \boldsymbol{h}_{t-1}) + \boldsymbol{b}_h) \tag{3.60}$$

The lookback parameter τ, also known as the enrol window, determines the number of past input vectors to use at each time step in recurrent units. This parameter is chosen according to the sampling rate and desired lookback duration. As an example, if the dataset sampling rate is 10 minutes and the desired lookback duration is 3 hours, the lookback parameter must be set equal to $3 \cdot 60/10 = 18$. The lookback duration should be chosen according to some information regarding the underlying system, or an appropriately long duration to capture the short-term variation patterns. Recurrent networks may be stateful, which means the state is kept between individual batches. In some cases with a small lookback parameter, statefulness may be required to ensure model convergence[48]. The backpropagation algorithm and optimization algorithms for training recurrent networks are extensions of those discussed in sections 3.3.5 and 3.3.6. It is beyond the scope of this thesis to derive and present the exact formulas or schemes.

3.3.8 Autoencoders

Autoencoders are neural networks using the same data as input and output. An autoencoder is trained to copy its input to its output, although designed not to do so perfectly[45]. The middle layer is commonly referred to as the encoding layer. If the size of the encoding layer is smaller than the input and output layers, the model is called undercomplete. An undercomplete autoencoder is similar to dimensionality reduction or feature extraction. By representing and reconstructing the input according to a reduced set of parameters, the most significant features may be acquired. If the encoding layer is wider than the input and output data, the autoencoder is called overcomplete. In this case, the use of regularization is particularly important. Otherwise, the network may learn to produce its output using linear relations to its input while keeping a large number of weights equal to zero. The same challenges as for traditional neural networks still apply for autoencoders, such as overfitting, underfitting and regularization.

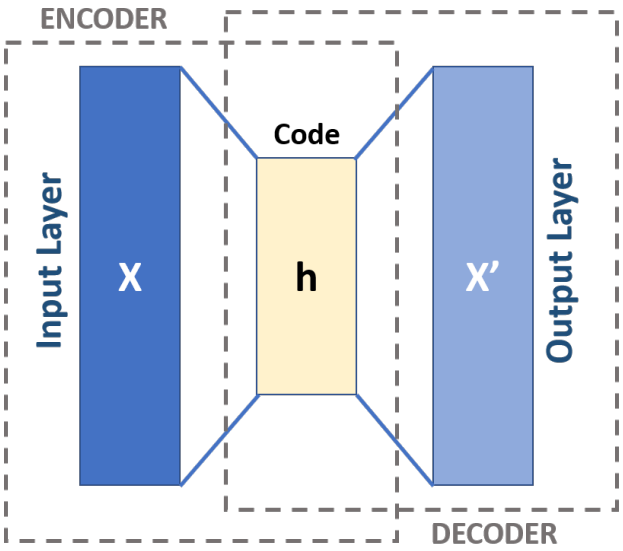


Figure 3.18: Illustration of autoencoder in which the middle layer is smaller than the input and output layers, with the encoder and decoder portion of the autoencoder outlined [63]

Figure 3.18 shows the general architecture of an autoencoder. An encoder, consisting of the layers from the input to the encoding layer, finds an encoding $\boldsymbol{h} = f(\boldsymbol{x})$. Meanwhile, a decoder consisting of the layers from the encoding layer to the output finds a reconstruction $\boldsymbol{r} = g(\boldsymbol{h})$. A loss metric $L(\boldsymbol{x}, g(f(\boldsymbol{x})))$ may be used to calculate the reconstruction error of the autoencoder. Both this reconstruction error and the error in individual outputs may be used to evaluate new data samples. If the decoder is linear and the MSE loss metric is used, an undercomplete autoencoder is essentially the same as Principal Component Analysis. Thus, nonlinear autoencoders can learn more powerful representations than PCA[45].

3.3.9 Ensemble models

Ensemble models look to combine the strengths of a collection of simpler base models[49]. Doing so may result in models with a larger hypothesis space and greater representative capabilities. The primary concern is which base learners to combine and how to combine them. Models may be combined through the use of an weighted average with weights learned on validation data, or using the model outputs as input to a new model such as a linear regression[48]. Ensemble models are typically used with fast algorithms like tree-based models to combine a large number of simpler models. However, fewer models may also be combined, like a selection of MLP or RNN models.

Because the error surface of neural network models often contain many local minimas or plateaus, models trained on the same data may benefit from the use of ensemble models. Most models used for state-of-the-art machine learning models in competitions, e.g. on Kaggle[64], use very large ensemble models[48].

3.3.10 Additional models

To benchmark the acquired results obtained in this thesis, a number of alternative regression models are used without detailed explanations. These include support vector machines, various regression models, decision trees, and random forest algorithms. A very brief introduction to each model is given.

Tree-based models define non-overlapping sections in the feature space, much like a series of if-else questions, giving the appearance of a tree. Such models may be desired due to simplicity, interpretability and relative robustness to outliers. However, tree-based models have shown to be unstable and provide poor accuracy compared to other model[44]. An improved variation of the simple tree-based model are Random Forest models. Such models reduce the model variance by averaging a large number of tree estimators.

Bagging regression fits basic regression models on several random subsets of data. Predictions are performed by aggregating the individual models, either by the use of averaging or voting among their predictions. Adaptive boosting regression works similarly, although models are trained on the same dataset with weights adjusted according to the error. This means it follows a gradient boosting scheme with corresponding learning rate and loss function. Any base estimator may be used, such as a simple decision tree. Elastic net is a regression method which combines LASSO and ridge regularization. Support vector machines fit a set of hyperplanes whose functional margin to the dataset points is maximized. The functional margin may be through of as the distance between data points and a decision boundary. Kernel function expansion may be used similarly as for linear regression.

3.3.11 Deep learning

Deep learning focuses on learning successive layers of increasingly meaningful representations in an automated manner[48]. It originally garnered additional interest in industry because it provided a scalable way of training nonlinear models on large datasets[45]. By adding depth and width to neural network layers, the ability of the model to represent complex functions increases. In many circumstances, using deeper models can reduce the amount of generalization error[45]. Deep learning has made extraordinary progress in the recent past[48].

Generally speaking, it is better to have too many hidden units than too few[49]. A neural network with too few hidden units will not be able to capture the required complexity of the underlying function. Deeper and wider neural network models are capable of representing more complex functions with high degree of generalization than shallow structures[45]. However, implementations in practice show that when little training data is available, it is preferable to use a small network with few hidden layers in order to avoid severe overfitting[48]. The number of layers and units should not be over-exaggerated, as problems like the vanishing gradient problem may occur. This implies that deep architectures should be used when a substantial amount of data is available.

Despite typically requiring large amounts of data, there are techniques which may enable the use of deep architectures despite lack of data. In tasks like image recognition or voice recognition, data augmentation is used to increase the number of training by performing transformations on data. Rotations, changes in pitch, changes in contrast and more may be applied, increasing the size of the dataset without needing to gather new data. Similar techniques for regression tasks exist, although typically not as powerful as for the former tasks. For example, random Gaussian noise may be added to increase model robustness. Alternatively, the same training data can be used several times by randomly shuffling the training data and thus training on different subsamples each time.

3.3.12 Hyperparameter selection and model fitting

The No free lunch theorem[65] states that any two optimization algorithm are equivalent when their performance is averaged across all possible problems. For machine learning in particular, this means there exists just as many problems for which some learning algorithm A beats another learning algorithm B on average as vice versa. Hence, it is impossible to find a single learning algorithm or architecture which is optimal for every problem. Therefore, the choice of learner and hyperparameters must be adjusted for each problem. Goals, error metric, target value and other hyperparameters should be chosen based on the problem that the application is intended to solve[45].

It is important to note that when fitting any machine learning model to a set of data samples, hypotheses are made that the output can be predicted given

the input, and that the available data provides sufficient information to learn the relationship between input and output[48]. If these hypotheses do not hold, the algorithm may fit a model which performs well on the provided data set, but does not necessarily represent any real functional relationship between the input and output. As a trivial example, imagine a linear model fitted to a set of input and output data known from theory to have a quadratic relation. A linear estimator for a quadratic function may be sufficiently accurate on a small interval, however it fails to capture the general trends of the quadratic function. Additionally, when training on past data to predict the future, an assumption is made that the future will behave like the past[48]. This assumption is not accurate for nonstationary problems, of which there are many in practice.

There are no formal rules for choosing optimal hyperparameters, meaning most choices must be evaluated empirically[48]. The design of hidden units is an extremely active area of research and does not yet have many definitive guiding theoretical principles[45]. Hyperparameters may be tuned manually or automatically. Automatic schemes are often computationally expensive, requiring some form of extensive search in the space of possible parameters. Some examples are grid search or random search. Manual parameter tuning means adjusting the hyperparameters one or two at the time in order to investigate which combination ensures the best model fit.

A standard starting point for every neural network is to use some form of mild regularization, as well as the Adam optimizer and ReLU activation function[45]. Another alternative is to develop a model that overfits, ensuring that it has sufficient capacity to model the desired problem[48]. Following this, appropriate actions can be made to avoid the overfitting of the model. Simple models should always be tried before complex ones, to justify the additional expense of complex models[48]. In recurrent layers, activation functions may be applied both for the internal neuron states and the neuron output. In the traditional implementation of LSTM and GRU, the sigmoid activation is used. Either ReLU or Tanh is typically used as output activation in hidden layers.

Validation

When training machine learning algorithms, available data is usually split into training, validation and testing sets, with a typically distribution of about 60, 20 and 20 percent, respectively. Training data is required to fit the machine learning model. Validation data is desirable to determine whether the model is able to generalize for new samples during training. Testing data may be required to ensure the final model still generalizes to new samples not in the validation set. All these sets should come from samples whose performance is considered as optimal or normal, thus enabling the use of the trained model on new data samples to find potential faults. For time series data, splits are often chosen sequentially, or samples for each split are drawn randomly.

An important practical consideration is that models can also be overfitted to the validation data[48]. This type of overfitting is achieved by gradually tuning hyperparameters to achieve a better model performance on the validation data, causing information to leak from the validation data into the model. When this is done at scale, an artificial level of performance is obtained. This emphasises the need for testing data, rather than just training and validation data.

Cross validation is often used as a validation scheme when sample size is limited. Popular variations are K-fold and leave-one-out cross validation. The former divides the data into K folds, training on $K-1$ folds while validation on the last fold, repeated for each combination of folds. This ensures all data samples are used for both training and validation. However, it also increases the time complexity by a factor of K. Leave-one-out cross validation is simply K-fold where $K = 1$, meaning one sample of data is left out for each iteration. Cross validation for trained models such as neural networks is not as easily implemented as for fitted models like linear regressions.

The loss metrics used when fitting models such as $RMSE$, MSE and MAE may also be used to evaluate trained models on new data. Additionally, coefficient of determination is often used to evaluate predictive performance. This metric is commonly referred to as R^2. It relates the performance of a model with predictive function f to the performance of a predictive model which always predicts the data mean $\bar{y}$:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} = 1 - \frac{\sum_i (y_i - f_i)^2}{\sum_i (y_i - \bar{y})^2} \tag{3.61}$$

SS_{res} is the residual sum of squares. SS_{tot} is the total sum of squares. R^2 is equal to 1 for a perfect model, and may be negative if the predictive performance is worse than that of the mean predictor.

Model callbacks

Callbacks are functions which may be applied during the training procedure to interfere with or alter the model training. Some callbacks, such as returning the calculated metrics, are performed automatically for each iteration. Other callbacks can be specified to interfere when certain criteria are met. Popular callbacks are early stopping, model checkpoints and learning rate scheduling.

Early stopping terminates the training process when a selected metric has failed to improve for a set number of epochs, typically the validation loss. This ensures the weights for which the minimum validation loss was calculated are kept in the finalized model, and thus used for predictions. An illustration of the optimal stopping point for a trained model can be seen in Figure 3.19. The number of epochs before training is stopped is referred to as patience. Early Stopping with a low patience restricts the trained model to a subspace of possible parameters relatively

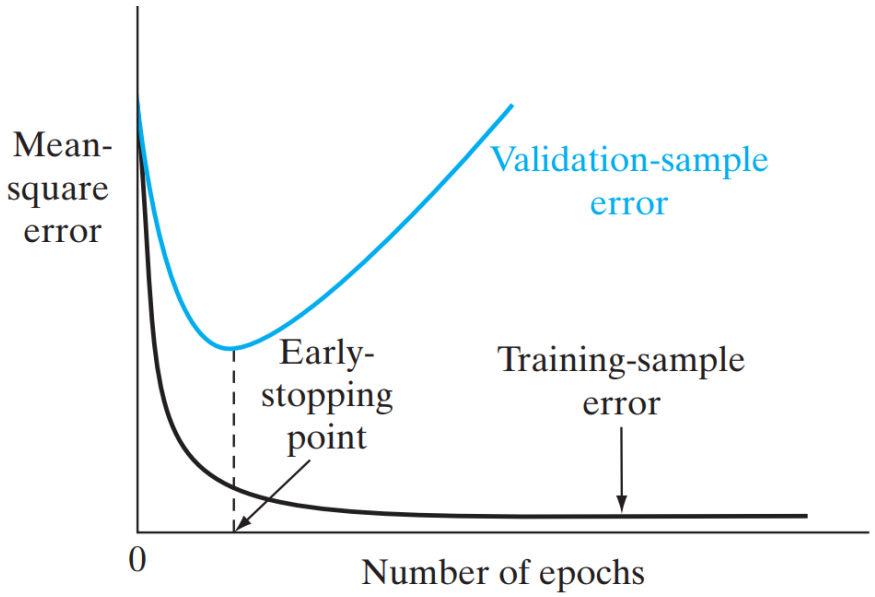


Figure 3.19: Illustration of early stopping [46, p. 174]

close to the initial parameter values. In the case of poor local minimas, this can be undesirable[46]. In most cases, early stopping acts as a form of regularization, not allowing the model to overfit.

Model checkpoints can be used to store weights and parameters during training. Doing so ensures the parameters obtained at any iteration throughout the training process can be loaded, regardless of which set of parameters were returned when the training algorithm terminated. Different sets of parameters can be loaded, for instance according to interesting points on the metric curve.

By the use of a learning rate scheduler, the learning rate can be adjusted during training according to desired, predefined values. For more advanced use cases, this can allow the training algorithm to avoid local minimas and converge to better solutions.

Weight initialization

Deep neural network models contain a large number of parameters, sometimes several million weights. The initial iterations during training may determine in which portion of the error surface the finalized model converges to. Therefore, initialization of parameters can be crucial in obtaining a desired model fit. Optimizing initialization strategies for neural networks is a difficult question, partly because neural network optimization in general is not yet well understood[45]. In particular, network units connected to the same input units must have different initial weights. Otherwise, the calculated output during the feedforward process and the gradient during backpropagation will be the same. In this case, the units

may only deviate from each other if one of the units is turned off at some iteration. To avoid identical weights for nearby units, weights are typically initialized using a Gaussian distribution with zero mean and small variance[45].

3.3.13 Uncertainty

Probability is an important underlying aspect of machine learning because any model must deal with the uncertain quantities from the system being modelled, the recorded data and the optimization algorithms used to train the models. Decision making in practice often requires uncertainty estimates, and many statistical preventive maintenance schemes rely on estimating remaining useful life.

With the use of dropout regularization, model uncertainty may be assessed as described by Gal and Ghahramani [15]. Performing a series of predictions with models using dropout gives a Gaussian distributed result, from which the model uncertainty may be evaluated. Note that this only addresses the model uncertainty. Uncertainty from other sources, such as measurement equipment, must also be accounted for.

In practice, the uncertainty revolving around the system or phenomenon itself may be equally important as the data and model uncertainty. For instance in the prediction of heat exchanger fouling, there is considerable uncertainty in evaluating how much fouling is acceptable before maintenance must be performed. Additionally, there may be uncertainty in how well the chosen predictive scheme approximates the fouling levels. Heat exchangers are typically delivered with specific fouling thresholds. Models using data and model uncertainty may calculate probabilities for when this threshold is exceeded, but whether the threshold itself is adequate is difficult to evaluate.

3.3.14 Time series data

Time series are sequences of data observed over a period of time[50]. When multiple variables are measured, the time series is referred to as multivariate. When data is measured at a defined sampling rate, the time series is discrete. Time series may consist of short-term, long-term and stochastic variations. These variations are combined as either additive or multiplicative terms, depending on the nature of the time series[50].

If statistical properties of the time series like mean and variance remain approximately equal over time, the time series is considered stationary. This is the case for time series without trends or seasonal components, which is normally not the case in practice[50]. Time series which are not stationary may be stabilized using differencing. Differencing can be seen as the equivalent of differentiating a function, calculating the change between consecutive samples rather than the sampled value in itself. This may be performed multiple times to eliminate higher-order trends.

3.3.15 Preprocessing

Before data is utilized in any machine learning algorithm, it must be preprocessed. This includes the removal of unwanted samples, as well as adjusting the data values to meet desired statistical properties. Most importantly, missing values must be replaced, or the corresponding dataset row removed. Otherwise, any calculations performed on the data are likely to cause errors.

Raw data from physical measurement systems often contain a substantial number of outliers that must be removed prior to or accounted for when fitting models. Outliers may be naturally occurring as part of the underlying process, or the result of faulty measurements. Ways of detecting outliers are by inspection using data visualization schemes such as box plots, scatter plots and regular time series plots for each feature. Outliers may be removed for each feature by defining ad hoc limitations or quantiles. Because simply removing outliers may be infeasible for certain methods, outliers may also be eliminated by interpolating nearby samples. Alternative approaches are clipping, for which values are clipped when exceeding a defined level, or winsorization, which limits values based on specified data percentiles.

For regression models, each input feature should be processed so that its mean value, averaged over the entire training sample, is close to zero. Additionally, the variance of all features should be approximately equal, and input parameters should be uncorrelated[46]. A widespread best practice is to do feature-wise standardization[48], as shown in Eq. 3.62. $\boldsymbol{\mu}$ is the vector of feature means and $\boldsymbol{\sigma}$ is the vector of feature standard deviations obtained from the training dataset. The equation is applied for each sample $\boldsymbol{x}$.

$$\boldsymbol{x}_{new} = \frac{\boldsymbol{x} - \boldsymbol{\mu}}{\boldsymbol{\sigma}} \tag{3.62}$$

An alternative to standardization is normalization, as seen in Eq. 3.63. Doing so transforms the data for each feature to a value range between 0 and 1. While some useful properties of normalization are strictly positive values and limited value range, this technique is very sensitive to outliers. Particularly large or small outliers which are not removed prior to normalizing the data may cause large areas of the value range to remain almost unused. As a result, standardization is typically preferred for regression problems[46].

$$\boldsymbol{x}_{new} = \frac{\boldsymbol{x} - x_{min}}{x_{max} - x_{min}} \tag{3.63}$$

Redundancy must often be removed from the data prior to model training. Otherwise, the same data may end up in both the training and validation data, for example if multiple rows of identical input and output parameters are present. It is generally safe to input missing values as zero, as long as zero is not already a

particularly meaningful value in the underlying system[48]. Thus, the use of zero in this manner may be unsuitable for boolean input parameters, but applicable for nominal data.

3.3.16 TensorFlow and Keras

TensorFlow is a system for implementation and deployment of large-scale machine learning models, with a focus on training and inference on deep neural networks[66]. It may be used in a wide range of devices, and offers model training on anything from single machines to clusters of thousands of CPUs or GPUs.

The ability to train deep models is particularly enhanced by the use of GPUs, due to their ability to parallelize applications. Neural network calculations largely consist of small matrix multiplications, for which parallelization is highly effective[48]. TensorFlow uses a core model dataflow graph which enables this at a large scale.

Keras is adopted as the high-level API of TensorFlow. It is perhaps the easiest deep learning framework to get started with[48]. Among its key features are that the same code can run seamlessly on both CPU and GPU, and that its API allows for quick prototyping of a large variety of machine learning algorithms[48].

Chapter 4

Methodology

4.1 Monitoring and sensors

To reduce the risk of overfitting and to save costs by installing and monitoring a reduced number of sensors, feature selection is desirable. However, the selected features must maintain enough information to provide sufficient monitoring of the system using the chosen methods. If the implemented monitoring scheme is not adequate for predicting certain faults, new sensors or equipment may have to be installed at the facility after production has started. The costs of doing so may exceed those of installing and monitoring these sensors from the get-go. A broad selection of sensors allows for the monitoring scheme to be altered, replaced or supplemented as deemed necessary. A balance between feature selection and monitoring flexibility must be determined. The chosen methodology should enable monitoring of existing facilities, and determine required monitoring for future, potentially unmanned, facilities.

In determining feature relations, heat exchanger theory is reviewed and statistical analysis is applied for extracted data. As noted by Guyon and Elisseeff [3], high variable correlation does not mean absence of variable complementary. Therefore, features cannot be selected or discarded according to correlation analysis alone. Hence, the underlying system physics must also be considered.

Review of existing facilities show that monitoring varies greatly. A summary of the general sensor availability can be seen in Table 4.1. Measurements related to the condition of the process fluid, namely process flow rate $\dot{m}_h$, process outlet temperature $T_{h,o}$ and coolant valve opening $v_{\%}$, are always available. Process flow rate must be measured to determine the facility production rate. Process outlet temperature is measured and kept at a certain setpoint using the coolant valve opening to control the coolant flow rate. The coolant fluid is subject to less monitoring in general. Three or more temperature measurements are usually available, while pressure measurements are more infrequent.

Sensor	Availability	Explanation
$T_{h,i}$	Often	Process inlet temperature. Measured at heat exchanger inlet or in relation to upstream equipment
$T_{c,i}$	Often	Coolant inlet temperature. Measured at heat exchanger inlet or in relation to central coolant distribution system
$T_{h,o}$	Always	Process outlet temperature. Measured at heat exchanger outlet or in relation to downstream equipment
$T_{c,o}$	Often	Coolant outlet temperture. Measured at heat exchanger outlet or in relation to downstream equipment
$P_{h,i}$	Sometimes	Process inlet pressure. Measured at heat exchanger inlet or in relation to upstream equipment
$P_{c,i}$	Sometimes	Coolant inlet pressure. Measured at heat exchanger inlet or in relation to central coolant distribution system
$P_{h,o}$	Sometimes	Process outlet pressure. Measured at heat exchanger outlet or in relation to downstream equipment
$P_{c,o}$	Rarely	Coolant outlet pressure. Measured at heat exchanger outlet or in relation to downstream equipment
dP_h	Sometimes	Process pressure difference. Installed or not based on the maximum allowed pressure difference in the exchanger
dP_c	Sometimes	Coolant pressure difference. Installed or not based on the maximum allowed pressure difference in the exchanger
$\dot{m}_h$	Always	Process mass flow rate. Measured in relation to upstream or downstream equipment. Note that the individual flows through heat exchangers in parallel are rarely measured, only the total flow
$\dot{m}_c$	Rarely	Coolant mass flow rate. Measured before, in or after in the heat exchanger
$v_{\%}$	Always	Coolant valve opening. Measured in relation to the heat exchanger

Table 4.1: Heat exchanger sensors

4.2 Fouling indicators

4.2.1 Facilities

Because the fouling level for real processing facilities cannot be measured directly, system conditions are traditionally derived using other estimates obtained through thermodynamic models as described in section 3.2.5. This may require extensive monitoring and several assumptions regarding the fluid properties. Even when such calculations can be performed, the results are only estimates, from which it can be challenging to determine the corresponding fouling level. This means accurate fouling estimates for evaluation of machine learning algorithms, and potential classification labels, cannot be obtained for real processing facilities.

To facilitate evaluation of the proposed fouling indicators for a controlled and deterministic fouling environment, use of simulated is proposed. Because fouling factor in itself is immeasurable in practice, using simulated data is the only way of comparing predictive results with factual data. During system simulations using appropriate process modeling software, the fouling factor R_f of a heat exchanger module may be set explicitly. Fouling indicators can be applied and evaluated for real facilities based on their ability to predict the added level of fouling for simulated datasets. For this reason, datasets obtained both through simulations and gathered from real facilities are used throughout this thesis. The facilities are discussed in detail in Chapter 6.

4.2.2 Selection of parameters

In theory, any combination of input and output parameters can be used in the implemented machine learning algorithms. Because testing and evaluating the predictive performance of every combination is infeasible, a systematic approach is necessary to determine suitable output parameters whose deviation between predicted and measured value indicate the level of fouling. Additionally, a set of input parameters capable of predicting these outputs must be determined. For simulated datasets, the resulting predictive model should produce a function $f(\boldsymbol{x})$ whose deviation $\boldsymbol{y}_t - f(\boldsymbol{x}_t)$ is correlated, and preferably linear, with the added fouling $R_f(t)$ at each timestamp t for input $\boldsymbol{x}$ and output measurement $\boldsymbol{y}$. The nature of this correlation may then be used to generalize the monitoring scheme for real facilities.

A common approach in machine learning is to predict system output parameters using system input parameters and any internal system parameters. For a heat exchanger in a fully monitored system, this would mean using the inlet temperatures, inlet pressures, pressure drops, compositions, flow rates and valve opening as input parameters to predict the outlet temperatures and pressures. More often than not, many of these measurements are not available. Additionally, several of these features may be excess, and a limited number of sensors is desirable. A more sophisticated approach to parameter selection is required.

In chapter 3, it was explained that methods using either pressure differences or temperatures and flow rates are most commonly used to estimate heat exchanger performance. As a reminder, pressure difference increases when flow velocity increases due to reduced cross section area or other changes in flow conditions, while heat exchanger thermal effectiveness decreases due to reduced heat conductivity. Combined with the sensor availability discussed in section 4.1, the subsequent fouling indicators in sections 4.2.3 and 4.2.4 are defined.

As described in section 3.3.12, there are no guarantees that the function $f(\boldsymbol{x}) = \boldsymbol{y}$ of a trained machine learning model for input $\boldsymbol{x}$ and output $\boldsymbol{y}$ represents a valid solution for the underlying real world problem. Therefore, theoretical argumentation is provided for each suggested estimator as to why machine learning algorithms may find an appropriate solution under certain assumptions. Unlike when using traditional thermodynamics modelling, the derived equations are not used directly, and approximations for underlying parameters are unnecessary. This simplifies the model definition, especially across facilities with different operating conditions and system designs.

4.2.3 Estimated Coolant Outlet Temperature

When fouling increases, thermal conductivity in the heat exchanger generally decreases. To maintain the required process outlet temperature given similar upstream and downstream operating conditions, the coolant inlet temperature must be reduced or the coolant flow rate must be increased. Decreasing the coolant inlet temperature is not feasible. Hence, the coolant flow rate must increase. When the coolant flow rate increases, the coolant outlet temperature decreases because each mol of coolant absorbs less heat. Therefore, increases in fouling should be noticeable in the form of decreasing coolant outlet temperature.

The equation for number of transfer units (Eq. 3.22) may be rewritten as follows:

$$\ln\left(\frac{E-1}{EC_r-1}\right) = \frac{UA(C_r-1)}{C_{min}} \tag{4.1}$$

$$\frac{E-1}{EC_r-1} = e^{\frac{UA(C_r-1)}{C_{min}}} \tag{4.2}$$

$$E = e^{\frac{UA(C_r-1)}{C_{min}}}(EC_r-1)+1 \tag{4.3}$$

Inserting the equation for heat exchanger effectiveness (Eq. 3.18), the following is obtained:

$$\frac{C_c\left(T_{c,o}-T_{c,i}\right)}{C_{min}\left(T_{h,i}-T_{c,i}\right)} = e^{\frac{UA(C_r-1)}{C_{min}}}\left(\frac{C_h\left(T_{h,i}-T_{h,o}\right)}{C_{min}\left(T_{h,i}-T_{c,i}\right)}C_r-1\right)+1 \tag{4.4}$$

Isolating $T_{c,o}$, the following relation is found:

$$T_{c,o} = \frac{\left(e^{\frac{UA(C_r-1)}{C_{min}}}\left(\frac{C_h\left(T_{h,i}-T_{h,o}\right)}{C_{min}\left(T_{h,i}-T_{c,i}\right)}C_r - 1\right) + 1\right) C_{min}\left(T_{h,i} - T_{c,i}\right)}{C_c} + T_{c,i} \tag{4.5}$$

Hence, the coolant outlet temperature can be written as a function f as follows:

$$T_{c,o} = f(U, A, C_{min}, C_r, C_h, C_c, T_{h,i}, T_{h,o}, T_{c,i}) \tag{4.6}$$

By introducing the following assumptions and writing $C = c_p\dot{m}$ where c_p is specific heat capacity at constant pressure, the number of parameters may be reduced accordingly:

- The learning method can model nonlinearities, or the expression $C_h - C_c$ does not change sign. Hence, C_{min} may be expressed as $C_{min} = f(C_h, C_c)$.
- The coolant flow is regulated according to the desired process outlet temperature. Hence, the coolant heat capacity C_c can be expressed as $C_c = f(c_{p,c}, \dot{m}_h, T_{h,o})$.
- The fluid compositions are constant. Hence, the specific heat capacity for each fluid is also constant.
- For the duration of the selected training data, the effects of fouling can be neglected both for the reduction in heat conductivity and cross section area. Hence, the heat transfer area A is unchanged while the overall heat transfer coefficient U is a function of the temperature differences and flows, $U = f(\dot{m}_h, T_{h,i}, T_{h,o}, T_{c,i}, T_{c,o})$.
- The equation f for U can be rearranged so that $T_{c,o}$ is isolated.

By eliminating excess parameters, the following relation is found:

$$T_{c,o} = f(\dot{m}_h, T_{h,i}, T_{h,o}, T_{c,i}) \tag{4.7}$$

When this function f is learned using a machine learning algorithm for data with minimal levels of fouling, predictions performed on future data using the trained model will assume no fouling is present. Hence, the prediction will be an expected coolant outlet temperature for a clean heat exchanger. When compared to target measurements in the presence of fouling, the predicted value should be higher than the measured value, because heat conductivity is lower than the model assumes. The challenge then becomes to relate the difference in temperature to the degree of fouling.

4.2.4 Estimated Pressure Difference

Some fouling phenomena, such as fouling in filters, have little effect on the heat conductivity. For such cases, the estimator described in section 4.2.3 is impractical. However, because flow is hindered, the pressure difference across the heat exchanger increases as a result. Assuming constant fluid composition and fluid density as a function of temperature, $\rho = f(T_i, T_o)$, prediction of the pressure drop using flow rate, inlet temperature and outlet temperature may be adequate. Hence, the following equations may be used:

$$dP_h = f(\dot{m}_h, T_{h,i}, T_{h,o}) \tag{4.8}$$

$$dP_c = f(\dot{m}_c, T_{c,i}, T_{c,o}) \tag{4.9}$$

Because the coolant flow rate is rarely measured in practice, the same assumption as in section 4.2.3 may be introduced if the coolant flow is controlled according to the process outlet temperature:

$$dP_c = f(\dot{m}_h, T_{h,i}, T_{h,o}, T_{c,i}) \tag{4.10}$$

Remembering that pipe pressure drop is a quadratic function of the flow velocity, pressure drop is largely correlated with mass flow rate. For normal operation with moderate levels of fouling, the increase in pressure drop due to fouling may end up being neglectable compared to differences due to changes in flow rates. If so, estimators relying on difference between predicted and measured pressure drop would be unreliable in practice. An exception is for occurrences of rapid or drastic fouling, for instance if heat exchanger channels are completely blocked or the the flow regime is otherwise greatly altered. In such cases, the increase in pressure difference is quite substantial. Accounting for cases of fouling in filters or drastic fouling encourages the inclusion of a pressure difference estimator, in addition to the temperature estimator discussed in section 4.2.3.

Note that the input parameters for dP_h and dP_c in Eq. 4.8 and Eq. 4.10 are equal to or a subset of the input parameters for $T_{c,o}$ in Eq. 4.7. This indicates that the three output parameters dP_c, dP_h and $T_{c,o}$ may be predicted using a single multioutput regression model. A disadvantage of using such multioutput models is that the model metric, for instance the mean squared error, is calculated based on all the output parameters. Hence, the model is trained to perform well for all output parameters rather than to perform optimally for a single output. Meanwhile, training and ensuring appropriate convergence for separate models may be time consuming and problematic. Once more, a compromise must be made to ensure the best predictive capabilities.

4.2.5 Predictive models

By including additional parameters such as coolant flow rate, valve opening and pressure sensors, the approximated function for each estimator should become more accurate. However, doing so increases the overfitting risk of the machine learning algorithms. Therefore, multiple models with a varying number of input parameters are defined. The combinations of input and output parameters can be seen in Table 4.2. Note that the term `model` in combination with a letter `A-E` refers to a `predictive model`, as opposed to a machine learning or regression model.

Multioutput models with two output parameters are used when measurements for dP_h are available. When not, only $T_{c,o}$ is used. dP_c is excluded as an output parameter because most facilities in practice lack measurement of the coolant flow rate. Although functional relations between coolant flow rate and other parameters are assumed according to Eq. 4.10, the accuracy of this approximation is not considered high enough to warrant the inclusion of this parameter given the disadvantages listed in section 4.2.4. The elementary input parameters are chosen according to Eq. 4.7. Supplementary models with the addition of $v_{\%}$, $\dot{m}_c$ and $P_{c,i}$ are defined to evaluate the effect of adding more parameters. The relation between each parameter is explored further in Chapter 6.

The parameter selection is based on the following hypotheses, which the thesis results will look to provide evidence in favor of or against:

> **Hypothesis 1**: The sensors $\dot{m}_h$, $T_{h,i}$, $T_{h,o}$, $T_{c,i}$ and $T_{c,o}$ suffice to accurately estimate heat exchanger fouling using nonlinear regression models, based on deviation between predicted and measured $T_{c,o}$. Additional sensors $v_{\%}$, $\dot{m}_c$ and $P_{c,i}$ may increase the model accuracy at the expense of higher overfitting risk and increased model complexity.

> **Hypothesis 2**: In the presence of heat exchanger fouling, deviation between predicted and measured dP_h may indicate whether fouling is on the process or coolant side. If such deviation is found, at least some fouling is on the process side. Otherwise, fouling is on the coolant side.

Name	Input	Output
Model A	$\dot{m}_h$, $T_{h,i}$, $T_{h,o}$, $T_{c,i}$	$T_{c,o}$, dP_h
Model B	$\dot{m}_h$, $T_{h,i}$, $T_{h,o}$, $T_{c,i}$, $v_{\%}$	$T_{c,o}$, dP_h
Model C	$\dot{m}_h$, $T_{h,i}$, $T_{h,o}$, $T_{c,i}$, $\dot{m}_c$	$T_{c,o}$, dP_h
Model D	$\dot{m}_h$, $T_{h,i}$, $T_{h,o}$, $T_{c,i}$, $v_{\%}$, $P_{c,i}$	$T_{c,o}$, dP_h
Model E	$\dot{m}_h$, $T_{h,i}$, $T_{h,o}$, $T_{c,i}$, $\dot{m}_c$, $v_{\%}$, $P_{c,i}$	$T_{c,o}$, dP_h

Table 4.2: Predictive models

4.3 Models and architectures

4.3.1 Evaluation metrics

Regression model performance is typically evaluated using metrics such as MSE, $RMSE$, MAE and R^2 on defined testing data, as explained in section 3.3. This is applicable if the training and testing time series data share the same statistical properties and underlying system characteristics. For many practical implementations, this is not necessarily the case.

A challenge when modelling heat exchanger data is that performance is continuously degrading towards an inevitable level of fouling, despite appropriate operating conditions. This level may not affect the system enough to warrant maintenance, but is nevertheless present. Hence, sequential selection of splits as explained in section 3.3.12 is infeasible, because the validation and testing sets towards the end of the time series will have different performance levels from the training set at the beginning. This difference in performance can only be neglected if at least one of the two following conditions are met. Firstly, if the sampling rate increases drastically, allowing large datasets to be gathered from short periods of operation, for which the performance decrease will be insignificant. Such high sampling rates may not be feasible or available. Secondly, if at the beginning of the selected training data, the heat exchanger has already experienced this level of inevitable fouling, after which it does not degrade significantly unless operating conditions change. Such specific data is difficult to find in practice.

An alternative method of selecting validation and testing samples is doing so randomly. This is not optimal either, because the choice of samples may have drastic effects on the validity of the trained models. The number of outliers selected in the validation and testing data compared to the training data may affect the model greatly. This effect is especially noticeable for small datasets, even if metrics not as affected by outliers like MAE are used. Additionally, data shuffling is typically not recommended for time series data[48]. Furthermore, using a significant portion of the available data for validation and testing reduces the number of training samples. Techniques like cross validation can counteract this issue, however these are not trivially implemented and evaluated for neural network models.

Based on the drawbacks of each scheme discussed above, training and validation sets are chosen randomly from data for which the heat exchanger performance is known or believed to be optimal or close to optimal. Performance is known for simulated data, while for real facilities it can be roughly estimated based on inspection of heat exchanger data and knowledge of maintenance dates. Separate testing sets are not used to validate the trained models. Model performance is primarily evaluated based on empirical interpretation of predictions performed on a portion of data not used for training or validation, in addition to calculated metrics like validation loss and coefficient of determination. An example of the chosen training, validation and testing scheme can be seen in Figure 4.1.

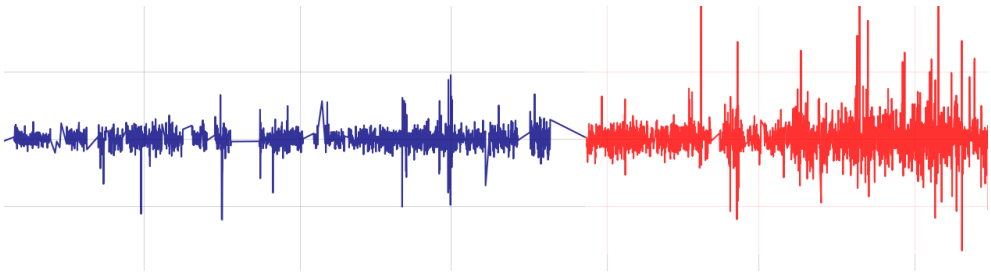


Figure 4.1: Example of time series data for an output feature. In blue is a series of points where performance is considered acceptable for model training. Training and validation samples are drawn from this portion of the data. Meanwhile, a portion of data with unknown performance levels is plotted in red. The trained models are used to predict feature values for this portion. Heat exchanger performance is evaluated by empirical analysis of the deviation between predicted and measured value, as well as various metrics.

Model weights are stored for the iteration which yields the minimum validation loss. The training loss for this iteration may not be equal to the minimum training loss across all iterations. Difference between the minimum and chosen training loss is an indication of the model's ability to overfit without appropriate measures.

Using empirical analysis to evaluate model performance has obvious drawbacks, as it does not generalize well for arbitrary processing facilities and requires extensive domain knowledge. However, with the shortcomings of simple metrics described throughout this section, and the otherwise large scope of this thesis, it is considered unproducable to implement more advanced evaluation metrics.

4.3.2 Unsupervised learning models

Correlation analysis and principal component analysis are applied to determine whether data visualization and statistical analysis can determine useful characteristics of heat exchanger data, and to determine suitable feature selection or feature extraction schemes. Statistical measures like standard deviation, skewness and kurtosis are analyzed for each dataset feature to determine the applicability of machine learning methods.

4.3.3 Supervised learning models

A selection of linear, multilayer perceptron and recurrent neural network regression models are implemented. Input and output parameters are chosen according to the predictive models defined in section 4.2.5. Performance is assessed as described in section 4.3.1.

Traditional linear regression models with built-in cross validation and ridge regularization are used due to simplicity and explainability. The performance of linear models provide a useful benchmark for more complex models, because simple

models are usually preferred if their results are satisfactory. Multilayer perceptron and recurrent neural network models are implemented based on the success of deep learning found in Chapter 2. More precisely, MLP, LSTM and GRU neural networks with one and two hidden layers and a wide range of hyperparameters are implemented, for which the most successful models are used to facilitate the thesis results. Ensemble models are implemented using linear, MLP and recurrent submodels. The output of each submodel is used as input feature to a linear regression model which calculates the ensemble model output.

4.3.4 Hyperparameter selection

Because performing grid searches over all possible combinations of hyperparameters is infeasible, some selected hyperparameters are common for all trained models used to produce the thesis results. Choice of parameters like optimizer, callbacks and batch sizes are largely based on recommendations from sections 2.2 and 3.3. Different alternatives have been tried for all hyperparameters, however some are tested more extensively through trial and error than others. Comparable performance is found between the RMSProp and Adam optimizers, as for the MSE (Eq. 3.31) and MAE (Eq. 3.33) error metrics. Appendix C.1 discusses error metrics for each facility. Appendix C.2.2 discusses various levels of LASSO and dropout regularization. Dropout regularization for LSTMs is discussed in Appendix C.3.1. Results for various MLP and LSTM architectures are presented in appendices C.2.3 and C.3.2, respectively. The selected hyperparameters are as follows:

- **Optimizer**: Adam
- **Error metric**: Mean Absolute Error
- **Learning rate**: Keras default (0.001 for Adam)
- **Batch size**: 256
- **Epochs**: Default of 500, although varied somewhat based on sample sizes. Number of epochs should be sufficiently high to ensure that model training is usually terminated by early stopping rather than the number of epochs.
- **Callbacks**: Learning rate reduction on plateaus is used to facilitate further learning if validation loss fails to improve for a number of consecutive iterations, 40 by default. Early stopping is used to terminate training when validation loss fails to improve for a number of consecutive iterations, 60 by default. Model checkpointing is used to store model parameters for the iteration with the minimum validation loss.
- **Regularization**: Dropout with a rate of 0.2. An equal amount of recurrent dropout is added for recurrent networks.
- **Layers and neurons**: Mostly one layer with 128 neurons. Otherwise, two layers with 64 neurons, or two layers with 128 neurons.
- **Lookback parameter**: Default of 12, equivalent to 6 hours with a sampling rate of 30 minutes.

4.3.5 Benchmarks

A benchmark which is easily implement, yet difficult to execute in practice, is the manual evaluation of dataset parameter plots. This is similar to how a supervising engineer would inspect a facility in real time, without the use of more complex methods or models. Evaluating the datasets in this manner, and comparing with the predictive results of the implemented models, provides insight as to whether the implemented models give any added benefits over existing monitoring techniques. Advice from Equinor employees is used as basis for the execution of this benchmark analysis, and is referred to as **manual parameter inspection**.

Some additional model benchmarks are implemented using models briefly explained in section 3.3.10. These include support vector regression, decision trees, random forest models, bagging regression and regression models with elastic net regularization. The models are chosen because finalized implementations are readily available in machine learning libraries, and can be used without much configuration. These benchmarks are referred to as **additional model benchmarks**.

It is important to note that while benchmark models are useful when evaluating model performance with a traditional scoring metric on defined testing data, it is not as straightforward when evaluating performance empirically. Limited time is put into understanding and tweaking the benchmark models, meaning it is inherently more difficult to evaluate these models empirically than the more well-understood, thoroughly tested models. Model benchmarking in itself is only feasible if the benchmarking models are known to perform relatively well, which is not the case for the detection of heat exchanger fouling. Hence, analyzing if the use of predictive models enhance the ability to detect fouling compared to manual and thermodynamic techniques is the primary concern.

4.3.6 Model training and hardware

A framework to facilitate rapid prototyping and comparison between different machine learning models is implemented. The implemented solution is described in Chapter 5, and available at `https://github.com/hermanwh/master-thesis`. Results obtained using this solution are demonstrated through a series of Jupyter Notebook files, found in the same code repository as the implemented solution. A brief description of each notebook is provided in section 5.2.3.

Ideally, the machine learning models would have been trained on a cluster of GPUs using Equinor's cloud solution, or locally on a desktop computer with a dedicated GPU. Powerful hardware facilitates the use of larger datasets, longer lookback duration and deeper models, as otherwise the algorithm run-times become overly lengthy and cumbersome to work with. Unfortunately, due to time constraints and various circumstances surrounding the COVID-19 pandemic, it proved difficult to obtain access to the desired computational resources. Because of data confidentiality, hardware and services outside of Equinor's domain could not be used.

As a result, all models used to obtain the thesis results are trained on an Intel Core i5-8350U notebook CPU, with 4 cores, 8 threads, a base clock of 1.7GHz and boost clock of 3.6GHz. This causes particular programs, specifically the ones comparing different recurrent neural network structures or predictive models for the largest datasets, to have run-times exceeding 24 hours. Most notebooks training MLP or recurrent networks have run-times between 30 minutes and 2 hours.

During development, experimentation with different architectures and hyperparameters was performed on a privately owned desktop running a dedicated Nvidia GeForce RTX 2060 Super GPU, with 2176 CUDA cores, a boost clock of 1815MHz, 8GB of GDDR6 SDRAM memory with a clock of 14000MHz, and CUDA Toolkit version 10.1.243. For this experimentation, various publicly available regression datasets were used rather than confidential Equinor data.

Chapter 5

Implementation

This chapter describes the implemented solution for the programming portion of this thesis. The chapter structure is similar to that of a software development thesis, although the content is naturally much more concise than for such a thesis. Understanding the technical implementation and its requirements is not necessary for reviewing the thesis results and discussion. However, anyone wanting to reproduce the obtained results, or experiment with the developed framework, are encouraged to read it in full. For more detailed documentation than provided in this chapter, the reader is referred to documentation in the code repository[67].

For the sake of clarity, some technical terms used throughout this chapter are as follows:

- **Variable**: a value used in code which may be changed during runtime
- **Argument**: a variable used as input to a function, as part of the function call
- **Parameter**: the input of a function, as part of the function definition
- **Function**: a section of code that takes a set of parameters, performs a specific task and returns a set of variables
- **Module**: piece of an individual program
- **Package**: collection of modules

5.1 Framework introduction

5.1.1 Intent

The Gas Production Systems department in Equinor for which this thesis is written has no predefined framework or general guidelines for machine learning research. Meanwhile, establishing connections to and familiarizing with the software systems of specialized data science departments within Equinor was considered too time consuming compared to the thesis duration. Therefore, it was decided to

develop the required framework for the thesis as a standalone project, with a preference for reusable and easily understandable code.

Implementing, analyzing and visualizing the performance of machine learning algorithms requires programming for tasks like preprocessing, building and training models, plotting, printing and more. During review of related work, it was noted that research projects focusing on machine learning often implement code as standalone notebooks or scripts, containing explicit code for these tasks in each document. While this approach provides a simple environment for prototyping, it makes large scale testing and comparison of different model configurations cumbersome by introducing potential boilerplate code. This suggests that modularization may be used to extract code for repeatedly used functionality, and encouraged the implementation of a top-level module. Because machine learning programming requires the use of many underlying frameworks, implementing a top-level module for commonly used tasks increases readability and usability for unfamiliar users by hiding the framework-specific implementation details.

The implemented solution intends to simplify, and thus encourage, continued research on the use of machine learning monitoring techniques. Despite this, generating interpretable and reliable results for the thesis itself has had top priority throughout. This means modular and reusable code comes as an addition to, and not at the expense of, the paramount thesis objectives.

5.1.2 Requirements

The requirements imposed by Equinor for the code implementation of this thesis are summarized as follows:

- Facilitate thesis results obtained through readable and runnable code examples, for instance using Jupyter Notebook.
- Persistent use of some source distribution control system, like Git.
- Not reveal or depend on any classified Equinor data or information.

In addition to the above requirements, some desirable features are as follows:

- Use of relevant and well-known programming languages and underlying frameworks for machine learning modelling, dataset processing, plotting and similar needs.
- Modular and reusable code, for instance in the form of a published code package.
- Usable and useful for both experienced and inexperienced machine learning developers.
- Facilitate accessible implementation of, and comparison between, different machine learning models and techniques.

5.2 Technical design

5.2.1 Implemented solution

The implemented solution uses Python[68] as programming language, with packages such as TensorFlow[66], Keras[69], Scikit-Learn[70], Matplotlib[71], Pandas[72], Seaborn[73] and Numpy[74] for the implementation, Git[75] for source distribution control and Jupyter Notebook[76] for exemplification. Its most important feature is that a large variety of linear models, neural networks, recurrent neural networks and ensemble models may be instantiated and used with very limited knowledge of the underlying implementation, by using the top-level stateful or stateless module. Differences between the stateful and stateless module are explained shortly. An example of a complete program using the stateful module, excluding metadata definitions, may look like this:

Code listing 5.1: Stateful module example

```
module.initDataframe(filename, columns, irrelevantColumns)
module.prepareDataframe(targetColumns)

linear = module.Linear("linear-model-test-1")
mlp_1 = module.MLP("mlp-test-1", [128])
mlp_2 = module.MLP("mlp-test-2", [128, 128])
ensemble = module.Ensemble("ensemble-test", [linear, mlp_1, mlp_2])

modelList = [linear, mlp_1, mlp_2, ensemble]

module.initModels(modelList)
module.trainModels()
module.predictWithModels()
```

Preprocessing is streamlined and simplified. After structuring a `.csv` or `.xls` data file and defining some metadata information like file path, desired names and units of each column, a list of columns to ignore, the list of output columns and the desired training and testing time period, preprocessing may be performed using the `initDataframe` and `prepareDataframe` functions.

Model instantiation for different architectures and arguments is made very simple. All model arguments are defined with usable defaults, so that the only required argument is a name chosen by the user. The above example shows the instantiation of a linear model, two neural networks with one and two hidden layers of 128 neurons, and an ensemble model of these, using the `linear`, `MLP` and `Ensemble` functions. Model training, prediction and plotting are performed using the `initModels`, `trainModels` and `predictWithModels` functions.

A stateful module uses an internal state to store variables. This means each method in the module can operate using a combination of user input arguments and stored state parameters. For the user, use of the stateful module means variables must never be supplied as arguments to multiple methods, and that variables returned from the module are never required as arguments for a module method.

The stateful module is useful for use-cases that follow the predefined schemes, or for users with limited programming experience. However, it can be cumbersome for advanced users, because it requires that methods are used in an appropriate order. Therefore, the stateless module offers the same methods as the stateful module, except all methods provide all its output variables and takes all its required input parameters directly from the function call. This is more useful when combined with traditional programming techniques like loops and functionality of the underlying frameworks. The same code example as seen previously for the stateful module becomes as follows when using the stateless module:

Code listing 5.2: Stateless module example

```
relevantCols, colDescs, colUnits, colNames, df = module.initDataframe(
    filename,
    columns,
    irrelevantColumns,
)
df_train, df_test = module.getTrainTestSplit(
    df,
    traintime,
    testtime,
)
X_train, y_train, X_test, y_test = module.getFeaturTargetSplit(
    df_train,
    df_test,
    targetColumns,
)

linear = module.Linear("linear-model-test-1")
mlp_1 = module.MLP("mlp-test-1", [128])
mlp_2 = module.MLP("mlp-test-2", [128, 128])
ensemble = module.Ensemble("ensemble-test", [linear, mlp_1, mlp_2])

modelList = [linear, mlp_1, mlp_2, ensemble]

maxEnrolWindow, indexColumn = module.initModels(
    modelList,
    df_test
)
module.trainModels(
    modelList,
    filename,
    targetColumns,
)
module.predictWithModels(
    modelList,
    X_train,
    y_train,
    X_test,
    y_test,
    targetColumns,
    indexColumn,
    columnDescriptions,
    columnUnits,
    traintime,
)
```

In the underlying implementation, an interface is defined to facilitate the usage of linear, tree-based, support vector, multilayer perceptron, recurrent, ensemble and autoencoder machine learning models in the same manner. The interface is implemented by different classes according to which models require different implementations. The interface definition is as follows:

Code listing 5.3: Machine learning model interface

```
interface MachinLearningModel():
    function init()
    function train()
    function predict()
    function save()
```

`Init` is called automatically when a class is instantiated. Each class is initiated with training data, underlying model architecture, a set of arguments and a user-defined name. Scalers used for preprocessing by standardization or normalization are automatically defined. The model architecture and arguments are later used for training and predictions. The provided name is used for storing models on a local storage device.

`Train` fits the model based on the provided data. Weight checkpoints are automatically used throughout. Recurrent models are trained using matrices and generators constructed based on the provided lookback parameter. Ensemble models are trained on the predicted results of their submodels using linear models.

`Predict` performs predictions using the trained model instance. Most models return matrices of predicted output values. Recurrent models using dropout regularization at predict time have the option of preforming multiple predictions, returning the matrix of the predicted values for each iteration, as well as the means and standard deviations across all iterations.

`Save` saves the architecture, trained weights and training history of a model on the local file system of the user. Saving and loading trained models is vital in ensuring research efficiency when running algorithms on low-end hardware such as personal laptops, because retraining each model used in a single script may take hours or even days.

Between the model classes and top-level module is a layer of functionality to perform operations such as printing, plotting, saving, loading, training, generate training summaries and various other utilities. Using this functionality directly, and combining it with use of the top-level module, provides the most flexible and powerful approach for experienced programmers. Most examples presented in section 5.2.3, from which the results of this thesis are generated, use additional functionality to enhance the top-level module.

5.2.2 Deployment

The code repository is available on GitHub[67]. The list of required packages and versions can be seen in the repository `requirements.txt` file. The associated `readme` file explains all necessary details regarding the repository. Additionally, the packaged project is published on the Python Package Index, so that it can be installed directly using the Python Package Installer `pip`. The package, named **HoWiML**, is available at `https://pypi.org/project/howiml/`. The corresponding GitHub repository is available at `https://github.com/hermanwh/howi-ml`. The implementation requires Python 3.6 due to version restrictions of underlying packages.

5.2.3 Demonstration

Use of both the high-level module and additional functionality is demonstrated through a series of Jupyter Notebook files. These are located at the top-level of the GitHub repository. All results presented in this thesis are available in these notebooks, which may either be read directly from GitHub, or run locally by forking the GitHub repository, installing the required packages and running Jupyter Notebook. Files are named using a prefix according to the following ordering:

- **0**: Dataset profiling
- **1**: Unsupervised and statistical methods
- **2**: Supervised methods using the top-level stateful module
- **3**: Supervised methods for more advanced implementations
- **4**: Supervised methods for experimental uses

The reader is referred to the notebooks for the practical demonstration. Data profiling is performed for each facility. Unsupervised methods include PCA and correlation analysis. Supervised methods include applying the predictive models A-E for each facility, as well as extensive comparison of network architectures, regularization parameters and loss metrics. Experimental use cases include performing predictions with models trained on different datasets, as well as model uncertainty assessment. Links to each individual notebook are provided as results are explored in chapters 6 and 7.

In general, results are reproducible if seeds and hashseeds are used, given that the neural network models are trained on CPU architecture. Models trained using GPUs are normally not reproducible due to non-deterministic behavior for parallel operations, and potential inaccuracies due to floating point precision. Results obtained for this thesis are not reproducible for the reader, as access to the data is restricted to Equinor personnel only.

Chapter 6

Facilities, data preprocessing and analysis

This chapter presents the facilities, acquired data and data preprocessing schemes used throughout the thesis work. In analysing the acquired data, a number of interesting relations are discovered. Some of these discoveries limit or affect how machine learning algorithms may be applied to the problem domain and how results should be interpreted. These findings are discussed in short throughout this chapter, and elaborated on in Chapter 8.

Data is obtained from facilities operated by Equinor. Data and tag names are classified, and thus the tag names used throughout this thesis are placeholder names based on the same naming scheme as for real tags. Because heat exchanger instrumentation is not standardized, the applicable monitoring schemes for each facility may vary.

The code repository used for data extraction and preprocessing is not made available to the reader due to access restrictions. However, some modified code extracts can be seen in Appendix B, detailing how the data is acquired and how preprocessing schemes are applied.

Note that the terms `dataset` and `facility` are used interchangeably when discussing the acquired data and results. For the sake of clarity, some statistical terms used throughout this chapter are as follows:

- **Kurtosis**: describes the shape of a probability function by specifying the length of the tails. Distributions with a small kurtosis value have fewer and less extreme outliers than the normal distribution.
- **Skewness** describes the shape of a probability function by specifying the asymmetry. Negative skew indicates a longer tail on the left.
- **Coefficient of variation (CV)**: ratio between the standard deviation and mean. Also known as relative standard deviation.

6.1 Facilities and heat exchangers

Facility D

As noted in section 4.2.1, exact fouling measurements can only be obtained from simulated data. For this reason, facility D is defined as a heat exchanger module with input and output fluid streams in the process simulation software UniSim. Two identical simulations are performed with fouling occurring on each side of the heat exchanger, resulting in two datasets. A fouling factor $R_f(t)$ is added directly to the heat exchanger module, equal to zero for the intended training data, and with two separate periods of linear increases in fouling during the testing data. The maximum added fouling factor is $12 \cdot 10^{-5} m^2 K/W$, which is somewhat lower than a realistic fouling threshold in practice. Fouling is reset, similarly as for a real heat exchanger maintenance wash, at the end of each of the two fouling periods.

A plate heat exchanger model with typical properties is used. The process fluid consists of mostly methane, with some heavier hydrocarbon components. The coolant is seawater. Inlet pressures are set as constant values. The inlet coolant temperature follows a sine pattern similar to how seawater would vary in practice, from January to June. Process inlet temperature and process flow rate are varied at predefined levels with added zero-mean Gaussian noise. Coolant flow is determined during simulations to obtain a constant desired process outlet temperature given the added level of fouling. Coolant outlet temperature, valve opening and process drop for each fluid are calculated as a result.

Because data is simulated, there are no outliers present in the resulting datasets. The datasets each contain 726 samples, of which 315 samples are used for training. The sampling rate is 6 hours. Note that this number of samples is rather low and the sampling rate is very high for use with deep learning methods. Sensors are shown in Table 6.1.

Table 6.1: Sensors, dataset D

Column	Description	Unit
20FT001	Process Flow Rate	Mg/hour
20PT001	Process Inlet Pressure	Bar
20PDT001	Process Pressure Difference	Bar
20TT001	Process Inlet Temperature	Degrees
20TT002	Process Outlet Temperature	Degrees
50FT001	Coolant Flow Rate	Mg/hour
50PT001	Coolant Inlet Pressure	Bar
50PDT001	Coolant Pressure Difference	Bar
50TT001	Coolant Inlet Temperature	Degrees
50TT002	Coolant Outlet Temperature	Degrees
50TV001	Coolant Valve Opening	Percentage

Facility F

Facility F is a heat exchanger at an offshore processing platform, at which a raw wellstream is separated into oil and gas. The coolant used is fresh water with antifreeze. Three identical heat exchangers are placed in parallel, referred to as F_a, F_b and F_c. Only two are used simultaneously in production. Following the heat exchanger, the process fluid enters a separator. This indicates that there will almost certainly be two-phase flow in the process portion of the heat exchanger.

Outlet conditions for the process fluid are specified to meet transport specifications for the gas. Process inlet and outlet temperature, as well as coolant outlet temperature and valve opening, are measured separately for each of the exchangers. Meanwhile, process flow rate and coolant inlet conditions are not measured separately, and so the process flow rate through each heat exchanger must be assumed equal. Additionally, the location of the process flow rate sensor is at the gas outlet of the separator following the heat exchangers. Thus, it is assumed that the fraction of gas and liquid remains stable. The exchanger design is PCHE, as described in section 3.2.1, with thinner channels than in traditional plate exchangers, increasing the fouling vulnerability.

Facility F has no previous history of fouling problems and no previous maintenance washes. However, operation was switched from using F_c to F_b in February 2019. If the data for each of these two heat exchangers is spliced at this point in time, the switch represents the equivalent of a heat exchanger wash. Despite lack of obvious fouling issues, the thermal conductivity should decrease over time. If the selected methods can detect an increase in performance following the switch of heat exchanger, this increase is likely to have occurred due to reduced fouling following the wash. For facilities with limited fouling it is easier to define feasible training data, because the fouling level is likely to remain slowly increasing throughout normal production.

Sensors for the spliced dataset consisting of measurements from F_c prior to and F_b following February 2019 are shown in Table 6.2. The number of sensors is rather limited, and thus only predictive models A and B as defined in section 4.2.5 are applicable, without the use of dP_h as an output feature.

Table 6.2: Sensors, dataset F

Column	Description	Unit
FYN0111	Process Flow Rate	MSm^3/day
TT0106MAY	Process Inlet Temperature C	Degrees
TIC0105CAYX	Process Outlet Temperature C	Degrees
TIC0425CAYX	Coolant Inlet Temperature	Degrees
TT0653MAY	Coolant Outlet Temperature C	Degrees
TIC0105CAY	Coolant Valve Opening C	Percentage

Facility G

As for facility F, facility G is a heat exchanger at an offshore processing platform, placed prior to a gas and liquid separator. The hydrocarbon fluid is cooled and separated in multiple stages, for which the heat exchanger in question is the second stage. Monitoring is significantly improved for this heat exchanger, with measurements of coolant flow and pressure differences, among others. This makes all the predictive models defined in section 4.2.5 applicable. The list of sensors is shown in Table 6.3.

Unlike facility F, seawater provided by a centralized coolant distribution system is used as coolant in the heat exchanger. The distribution system supplies several different platform components. This means the coolant rate provided to the heat exchanger may depend on the conditions of other connected components, which are not measurable in relation to the heat exchanger itself. The use of seawater also means the coolant inlet temperature varies with a seasonal trend. Such facilities sometimes struggle to provide sufficient cooling during the summer when seawater temperatures are high.

Facility G has experienced fouling for an extended period of time. Maintenance was performed regularly between March 2017 and April 2019, with noticeable and continuous decreases in performance following each maintenance. Confirmed maintenance dates are shown in Table 6.4. Fouling is believed to occur primarily because of biological fouling in the coolant portion of the heat exchanger, due to cracks in the coolant system pipeline. Seawater coolant systems typically gather water from significantly below the ocean surface to avoid most marine life and organisms. Cracks cause water to be gathered closer to the surface. Additionally, cases of fouling on the process side of the heat exchanger are known to have occurred, although at a lesser extent.

Table 6.3: Sensors, dataset G

Column	Description	Unit
FI0010	Process Flow Rate	m^3/hour
PDI0064	Process Pressure Difference	Bar
TZI0012	Process Inlet Temperature	Degrees
TI0066	Process Outlet Temperature	Degrees
FI0027	Coolant Flow Rate	m^3/hour
PI0001	Coolant Inlet Pressure	Bar
PDT0024	Coolant Pressure Difference	Bar
TT0025	Coolant Inlet Temperature	Degrees
TT0026	Coolant Outlet Temperature C	Degrees
TIC0022U	Coolant Valve Opening C	Percentage

Table 6.4: Maintenance dates, dataset G

Month and year	Maintenance
03.2017	Traditional wash
09.2017	Traditional wash
01.2018	Acid wash
05.2018	Acid wash
08.2018	Filter change
10.2018	Acid wash
04.2019	Acid wash

Other facilities

Several additional facilities were considered for use during the thesis work, but were excluded for various reasons. To rationalize the apparently unsystematic facility naming scheme presented above, the facilities which were excluded are as follows:

- **Facility A/B**: Heat exchangers in operation at a facility operated by Equinor. Discarded as a possible use case due to lack of monitoring, most importantly no measurement of coolant outlet temperature.
- **Facility C**: Test rig at Equinor's research facility. Used to investigate the fouling phenomenon of an undisclosed substance for treatment of natural gas. Discarded due to findings by Horn [77], arguing that data from these tests are unsuitable for fouling prediction under regular operating conditions.
- **Facility E**: Heat exchanger in operation at a facility operated by Equinor. Discarded as a possible use case due to lack of known instances of fouling and maintenance history.

6.2 Data extraction

Data is extracted from Equinor servers using a Python API. The API allows extraction of data using a predefined list of tag names and a sampling rate. A modified code example can be seen in Listing B.1. An average of measured values according to the chosen sampling rate is returned. For sensors measuring values which are system setpoints, this occasionally leads to artificially constant values. Hence, high sampling rate is preferred when possible. High sampling rate also facilitates the use of deep machine learning methods, as it increases the sample size.

Due to the COVID-2019 pandemic, use of VPN, and limited computing power, it proved difficult to extract and use data with the desired sampling rate of 1 minute or less. As a result, sampling rates of 10 and 30 minutes are used for the extracted data.

6.3 Filtering and preprocessing

Every dataset row where one or more features have missing values is removed. While methods to replace missing values exist, either by interpolating nearby values or using more advanced methods like the Amelia algorithm[78], it is found that the obtained datasets contain very few missing values. When missing values are encountered, the same feature is typically missing for many consecutive rows. This makes handling of such values difficult, and replacing the values would introduce considerable uncertainty.

The raw data contains numerous outliers, most of which are difficult to determine whether are the result of real changes in operating conditions, or faulty measurements. The latter should ideally be removed, while the former should not. Detecting outliers for multivariate data is very difficult in general, because determining the nature of an outlier for a single feature may require comparisons with several other features. For large datasets, this approach to outlier removal is infeasible.

Removal of even the most obvious outliers is not trivial for data used in recurrent machine learning models. Considering that recurrent networks rely on the processing of sequential data, it is desirable that the time step between each sample is uniform. Meanwhile, metrics like the squared error often used in practice are sensitive to outliers. An option to remove outliers while maintaining uniform time steps is to use some form of interpolation scheme as mentioned briefly in section 3.3.15. However, this is far from trivial for multivariate data.

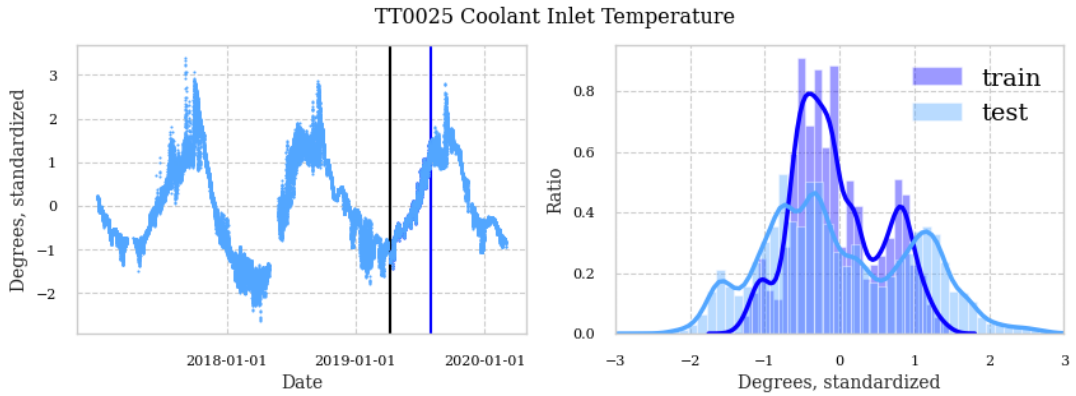
A compromise is made by removing the most apparent outliers, through inspection of reasonable cutoff values or quantiles for each feature. Modified code examples are seen in Listings B.2 and B.3. Assuming the number of removed outliers is small compared to the total number of samples, a limited number of sequences used to train recurrent networks are affected. A large majority of the removed outliers for all features come as a result of removing samples with zero or very low process flow rate compared to the moving average, indicating a halt in production at the facility. The nature of such halts mean several samples in a row are removed, further reducing the negative effects on sequential training. Outlier samples for other features are removed according to reasonable thresholds for each facility, defined in an ad hoc manner. There are no redundant rows present in the datasets after the removal of missing values and outliers.

Standardization (Eq. 3.62) is applied to obtain features with zero mean and equal variance. The mean and variance used for standardization of each feature are stored in the machine learning model based on the training data. Therefore, any testing data provided to a trained model will use the same standardization parameters. This is required, because otherwise the model would treat potentially different datasets as equal based on different standardization parameters. However, it does make model prediction on data from other facilities infeasible, unless the standardization parameters of the model are changed manually.

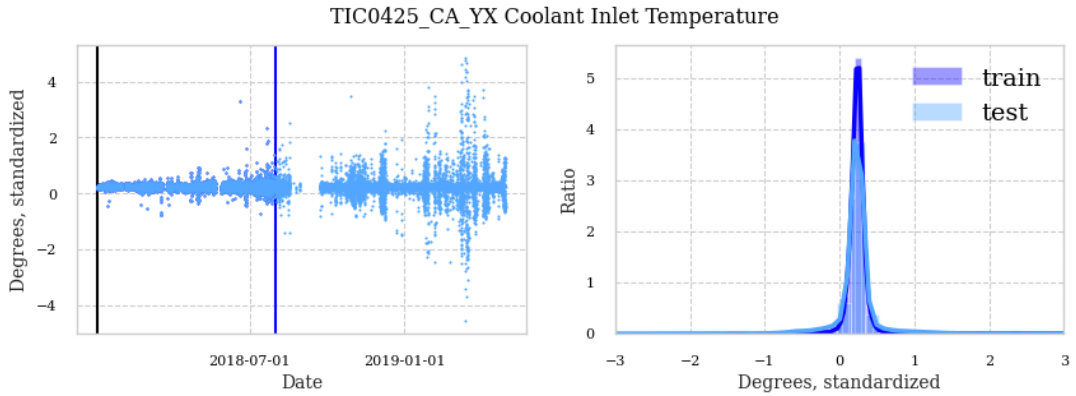
6.4 Data analysis and discoveries

This section analyses the data acquired from the previously mentioned facilities, in order to determine interesting properties and relations which may affect the applied machine learning methods. Tables E.1, E.2 and E.3 show relevant statistics such as mean, standard deviation, kurtosis and skewness for each dataset.

6.4.1 Coolant inlet temperature



(a) Example from facility G showing how coolant inlet temperature varies in a yearly cycle when seawater is used.



(b) Example from facility F showing coolant inlet temperature for a closed-loop system.

Figure 6.1

In Figure 6.1a, a graph of the coolant inlet temperature for facility G can be seen on the left-hand side. As mentioned, the facility uses seawater from a centralized distribution system as coolant, meaning the temperature varies with a seasonal pattern. Due to the nature of heat exchanger fouling, training periods are very unlikely to cover an entire year. Therefore, a necessary assumption when training models on heat exchanger data is that the algorithm is able to learn the seasonal functional variation despite only training for a limited time period. This is not

necessarily the case, because the total required coolant demand may be higher during periods with high coolant inlet temperature, affecting the operating conditions of several components. The sampling distributions on the right-hand side indicate that training data from a limited time period is likely to have shorter tails, and thus a smaller kurtosis value, than the overall distribution.

Figure 6.1b shows the coolant inlet temperature for a closed-loop coolant system. The temperature is stable under normal conditions, with much shorter tails than for seawater systems. However, because the coolant is recooled in a separate system, faults with this separate system may affect the inlet temperature of the coolant. This can be seen towards the end of the time series, where the number of outliers increase drastically. During this period, the facility is known to have had problems with the recooling system.

6.4.2 Coolant outlet temperature during fouling

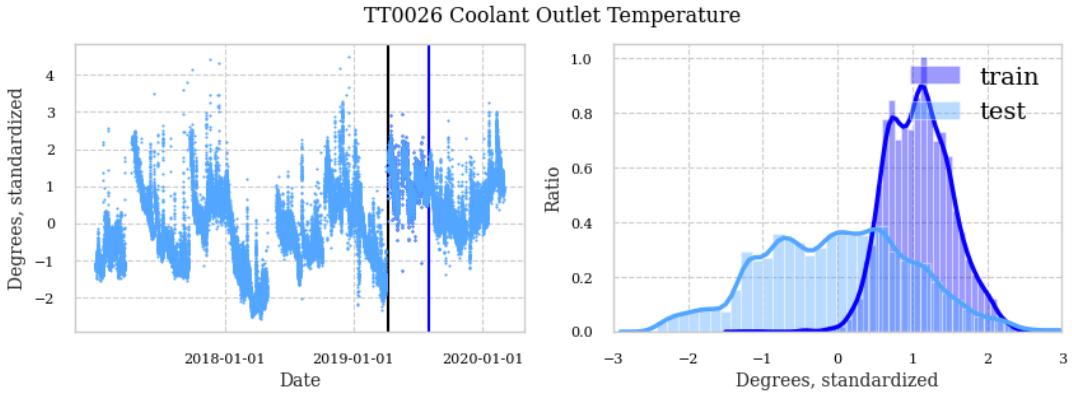
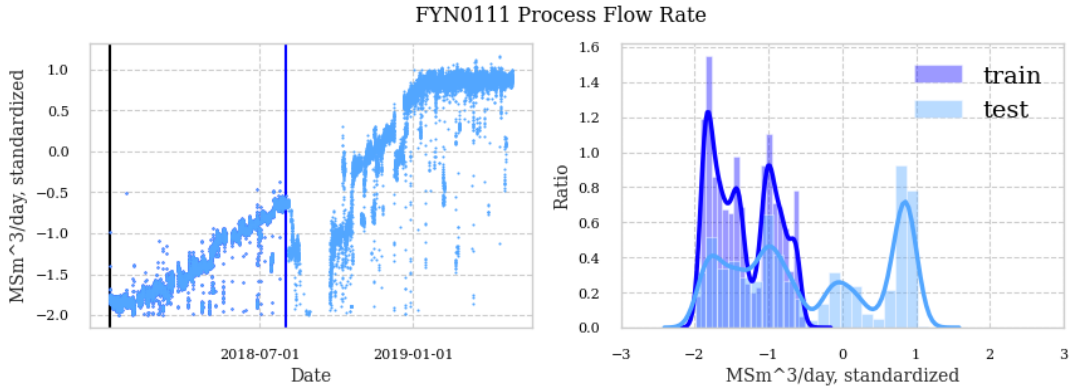
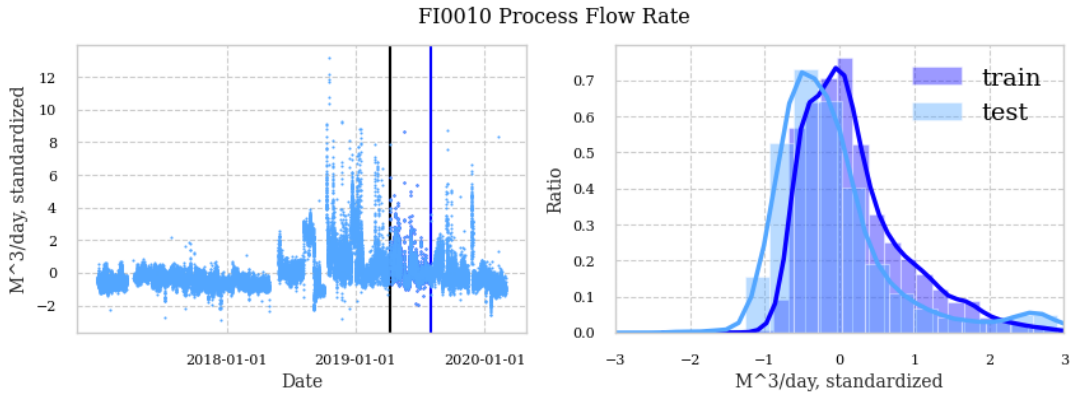


Figure 6.2: Example showing how the coolant outlet temperature decreases as fouling increases.

The left-hand side of Figure 6.2 shows the coolant outlet temperature of facility G over an extended period of time. It is clearly visible that the outlet temperature decreases over time, before substantial jumps are seen following each maintenance. In practice, such plots are one of the factors used in determining the need for heat exchanger cleaning. The analysis of such graphs, among others, is part of the **manual parameter inspection** benchmark defined in section 4.3.5. However, there may be reasons other than fouling for gradual decreases in outlet temperature, such as increases in production rate or changes to upstream operating conditions. Comparing the coolant outlet temperature graph to a predicted level as suggested in section 4.2.5 gives higher confidence in the source of the decrease. The sampling distribution on the right-hand side shows considerable difference between the selected training data and the overall distribution, with higher skewness and smaller kurtosis for the training data.



(a) Example showing gradual increases in process flow rate for facility F. Outlier measurements are primarily seen in the negative direction.



(b) Example showing stable process flow rate for facility G. Outlier measurements occur in both directions, however mostly in the positive direction.

Figure 6.3

6.4.3 Process flow rate

Figures 6.3a and 6.3b show the process flow rates for facility F and G, respectively. Clearly, the two facilities are at different stages of the production timeline. Facility F gradually increases the production rate until the designed maximum capacity is reached around the start of 2019. This gradual increase means predictive models must account for both the long-term linear increase due to increased production rate and the short-term variations due to system behavior. The sampling distributions are vastly different. Meanwhile, facility G has already reached its designed capacity before the start of the acquired time period. Hence, the training data has a similar sampling distribution as the overall dataset.

The process flow rate graphs also show vastly different outlier behavior. For facility F, almost all outliers for flow measurements are below the moving average. This is despite the removal of all readings with flow value below a certain limit, so that

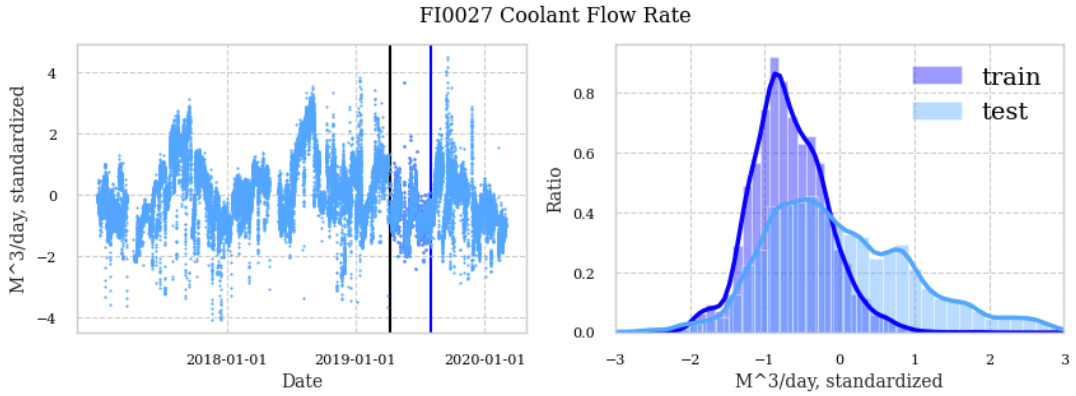
no cases of complete stops in production are contained in the dataset. Removing the remaining outliers is very difficult because a number of these readings are in fact realistic variations due to decreases production levels for some period of time. Meanwhile, outliers for facility G generally seem to be much less frequent, and in the positive direction compared to the moving average.

6.4.4 Coolant system correlation

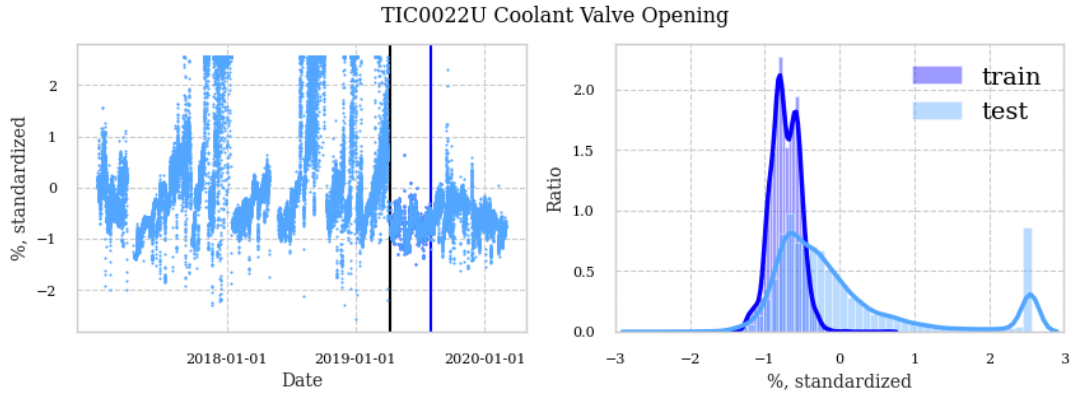
Figures 6.4a, 6.4b and 6.4c show the coolant flow rate, valve opening and pressure difference of facility G. These values should in theory be highly correlated. Coolant flow rate is controlled by the valve opening, while pressure drop is largely a function of the flow rate. As seen in the figure, the coolant flow rate is not as correlated with the valve opening as one would expect in an ideal system. For instance, as seen towards the end of 2017, the valve opening is increased to its maximum value while the flow rate continues to decrease. This indicates that there are other factors than the coolant valve opening which control the coolant availability, such as the coolant requirement of other components using the same coolant system. Additionally, the relationship between valve opening and flow is not linear in practice. It appears that increasing the valve opening past approximate 60% has little effect on the flow rate. This suggests linear models using valve opening to estimate coolant flow rate may yield unsatisfying results for large $v_{\%}$ values. Overall, the coolant valve opening does not seem to be a sufficient measurement to estimate the coolant flow reliably in the general case.

The pressure difference is increasing drastically during this same period. It is natural to assume that increases in both flow rate and fouling contribute to the increases in pressure drop. This indicates that comparing the pressure drop to the coolant flow could be an applicable fouling indicator in the case where coolant flow rate is measured. Even so, the increases are very drastic, and quickly plateaus when the real pressure differences exceed the sensor measurement limit. Therefore, it is difficult to estimate the fouling effect, especially on the thermal conductivity of the heat exchanger, using these plots. Thus, a predictive model which accounts for the pressure drop due to increased flow rate is more suitable for determining the effect of fouling than manual inspections.

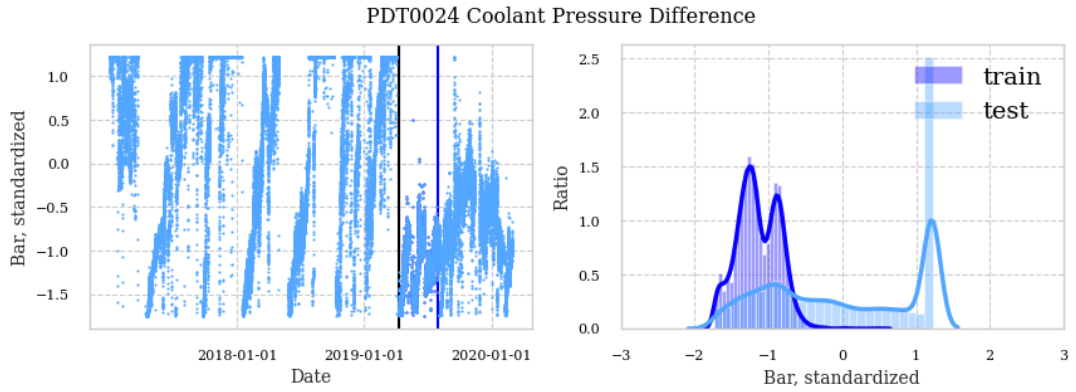
The rapid increases in coolant pressure drop indicate that fouling starts to accumulate once the heat exchanger starts operating. Therefore, these areas of operation are unsuitable as training data for machine learning algorithms. A period of more stable operation seems to occur following the last heat exchanger maintenance during spring 2019, for which the training period is defined. The acquired data spans several years, with only a small portion is used for training. Thus, the resulting models may not generalize for the entire testing period, because variables like upstream and downstream operating conditions, process flow composition and more could have changed.



(a) Coolant flow rate for facility G.



(b) Coolant valve opening for facility G.



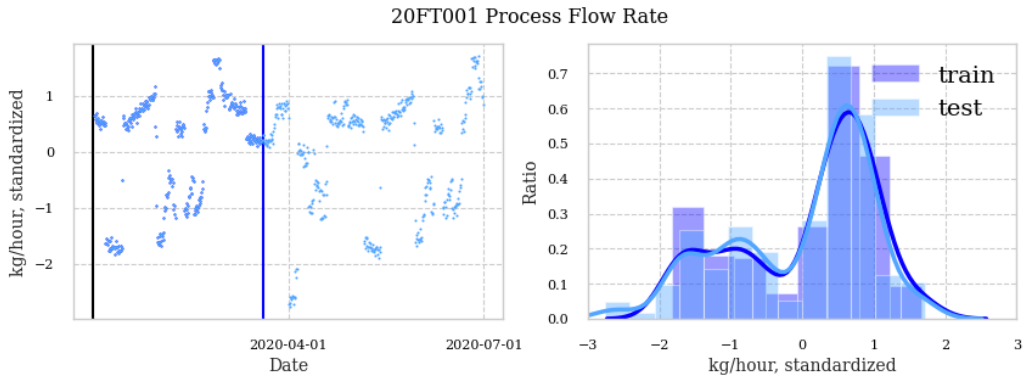
(c) Coolant pressure difference for facility G.

Figure 6.4

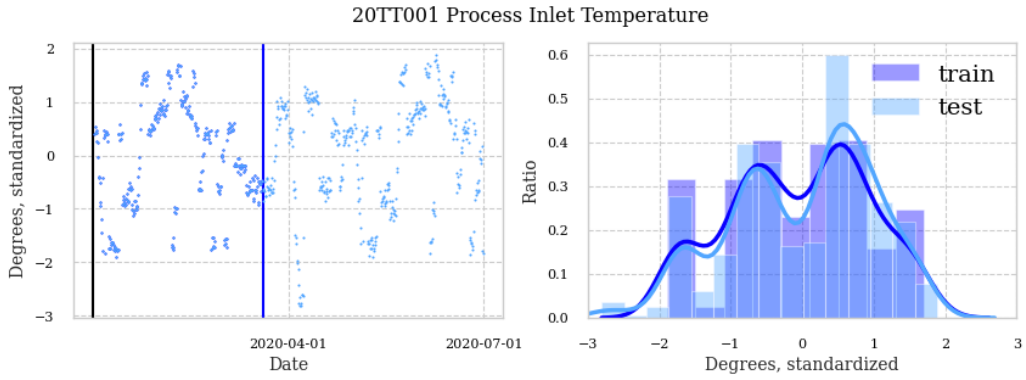
6.4.5 Variation in input data of process simulations

Figures 6.5a and 6.5b show the process flow rate and process inlet temperature for facility D. These are the input parameters with the most variation used during UniSim simulations. Remaining parameters are either constant or varied slightly, as described in section 6.1. Compared to the real facilities F and G, the process flow rate and inlet temperature are varied more rapidly and uncorrelated, to cover a large possible area of operation with few data samples.

Notice that the sampling distributions are almost identical for the training and testing data, unlike many of the sampling distributions seen previously. This illustrates the fact that simulated data have many desirable properties which may not be the case for real facilities. These properties, and the lack of outliers, are likely to yield increased accuracy for simple models like linear regression models. In general, models incapable of achieving high accuracy on simulated data with desirable statistical properties are unlikely to be useful for real data. An exception may be for deep learning models requiring large amounts of data, for which the size of the simulated dataset may not suffice.



(a) Process flow rate for facility D.



(b) Process inlet temperature for facility D.

Figure 6.5

6.4.6 Limitations of the process simulation models

The processing model is defined, and simulations are performed, by Equinor personnel with expertise within heat exchanger modelling and operation. When defining the processing model, it was proven difficult to model the changes in flow conditions due to fouling. The effects of heat exchanger fouling on pressure difference and cross section area are generally not well understood, and thus modelling this relation was determined to be infeasible. As a result, only thermal effects of added fouling are accounted for in the simulation model. Hence, fouling indicators relying on deviation between predicted and measured process pressure difference are unlikely to yield satisfying results for the simulated dataset.

The shortcomings of the simulation model affect the applicability of process pressure difference as an output feature for the remaining datasets, specifically dataset G. Because the thesis methodology relies on evaluating fouling indicators based on empirical analysis, determining the performance of such models is infeasible when no comparisons to known cases of fouling can be made. Furthermore, the use of multiple output features affect the training metrics of machine learning models, as discussed in section 4.2.4. Trained models may not be as capable of accurately predicting the coolant outlet temperature if additional output features are included. The combination of these drawbacks warrants increased emphasis on predictive models relying on thermal effects rather than hydraulic effects like pressure difference and flow conditions, and the training of individual models for each output parameter.

Chapter 7

Results and analysis

Results obtained using unsupervised methods are presented in section 7.1, with principal component analysis in section 7.1.1 and correlation analysis in section 7.1.2. Results for the supervised predictive models defined in section 4.2.5 are presented in section 7.2, with facilities D, F and G in sections 7.2.1, 7.2.2 and 7.2.3, respectively. The performance of linear, MLP, LSTM and benchmark regression models are evaluated. Some experimental results concerning predictions across facilities and predictions with model uncertainty are presented in sections 7.3 and C.4, respectively. Less emphasis is placed on the unsupervised and experimental results, in favor of evaluating the predictive models. The results are discussed further in Chapter 8.

Many plots and additional metrics are placed in Appendix D to keep the chapter concise. Because presenting all the produced plots, metrics and model convergence patterns is infeasible, the reader is referred to Jupyter Notebook implementations for additional details. Links to each notebook are provided as footnotes.

7.1 Unsupervised methods

7.1.1 Principal component analysis

Principal component analysis is performed to facilitate data visualization in two dimensions. Training and testing data are plotted with changes in color according to the time parameter, in order to investigate how the operation point in 2D changes throughout the time series. Training data is chosen so that two separate periods with believed increases in fouling are present. A Jupyter Notebook containing all plots and results can be seen on GitHub[1]. The produced plots are interpreted and evaluated empirically. Acquiring any decisive information about the principal components requires extensive analyse of each dataset feature.

[1] https://github.com/hermanwh/master-thesis/blob/master/1_pca.ipynb

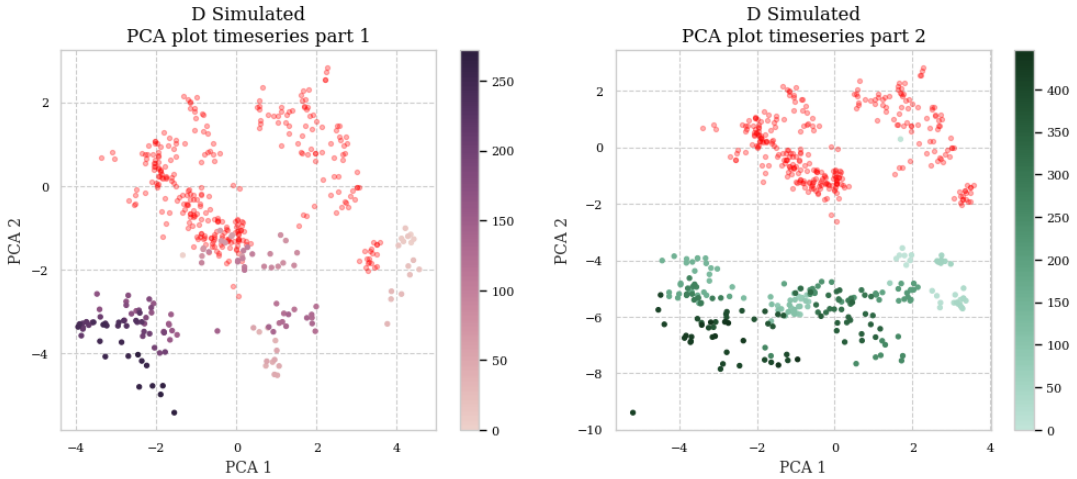
Figures 7.1a, 7.1b and 7.1c compare the changes in operating point for facilities D, F and G, respectively. For these plots, the color gradient scale seen on the right-hand side of each plot represents the index of each dataset row in the training data. The first period of testing data varies from light pink to dark purple, while the second period varies from light green to dark green. Because using two different color gradient scales in the same plot may be confusing, all training points are plotted in red when compared to testing data. This means changes in operating point for the training data cannot be seen in these plots. Figures D.1a, D.1b and D.1c show the PCA 2D plot for the training data of facility D, F and G, respectively. These are included in the appendix so that the changes in operating point for the training data may also be analyzed.

For facility D, training points are rather spread due to the rapid variations noted in section 6.4.5. For the first testing period of linearly increased fouling, the operating point appears to be moving downwards and to the left. Meanwhile, for the second period of fouling, the movement of the operating point has no clear pattern. All points are considerably further in the negative PCA2-direction than for the first period, suggesting that the operating point in general is quite different despite similar levels of fouling.

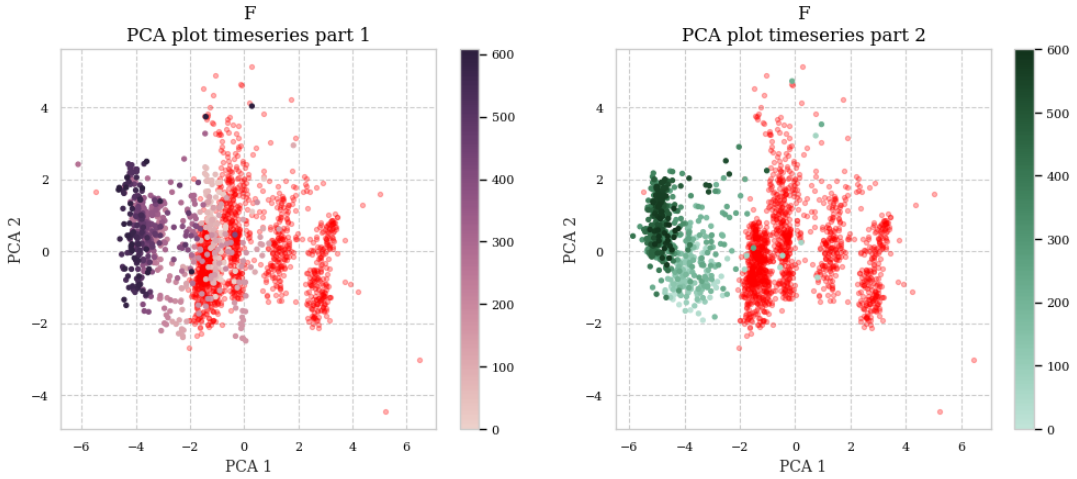
For facility F, there seems to be two clear directions of variations. The operating point is moving slowly towards the left, with more rapid variations in the vertical direction. Considering the linear increases in process flow rate for much of the time series for facility F, it appears logical that the PCA1-axis may correspond to this parameter. Following the heat exchanger maintenance separating part 1 and 2 of the time series, the operating point shifts slightly to the right, before once more slowly moving towards the left. This trend could be interpreted as gradual fouling in the heat exchanger, although this is only suggested and by no means validated by these findings.

For facility G, the same trends as for facility F can be seen during the first testing period, although in different directions. The operating point is trending downwards and to the right as the time series progresses. At the beginning of the second testing period, the operating point has changed significantly from the end of the first testing period. During the second testing period, the operating point hardly moves at all. If anything, the movement is slightly upwards, which differs significantly from the first period. There is no clear correlation between the two testing phases, and no clear relations to the PCA plots of previous datasets.

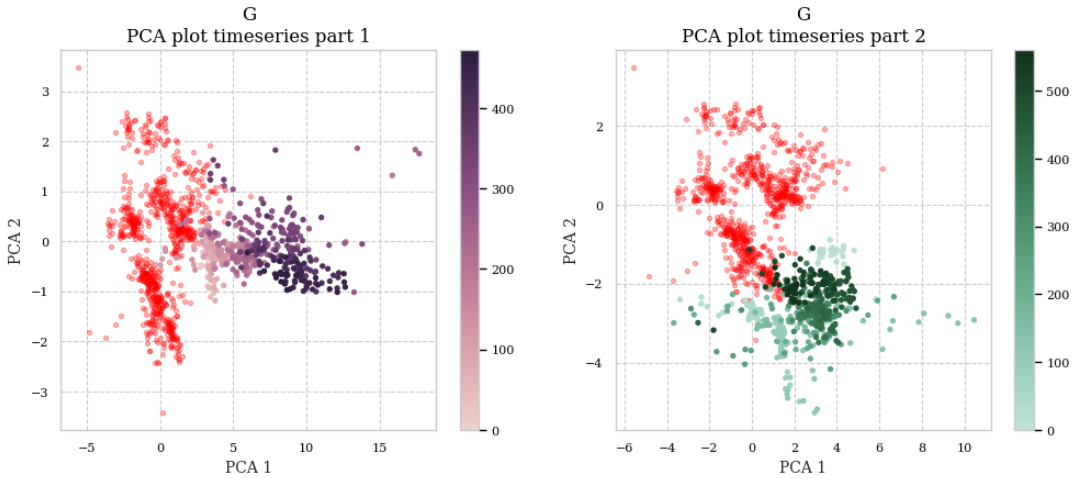
Attempts are made at using PCA models fitted on a single dataset to plot the operating point of several facilities, seen in Figures D.2a and D.2b. Some similar trends are found as for when each dataset is fitted and used with the same PCA model, with operating points moving in relatively comparable directions. However, without knowledge of the variation each PCA-axis represents, it appears difficult to identifying any type of fouling.



(a) Principal component analysis of two testing data periods in dataset D.



(b) Principal component analysis of two testing data periods in dataset F.



(c) Principal component analysis of two testing data periods in dataset G.

Figure 7.1

7.1.2 Correlation analysis

Analysis of the correlation matrix for each dataset is performed to discover possible changes in feature correlation when fouling is present in the heat exchanger. Results for facilities D and G are presented, as these contain the same sensor measurements, and are thus most suitable for comparisons. A Jupyter Notebook containing all plots and results can be seen on GitHub[2].

Figures D.3a and D.3b show the correlation matrices for facilities D and G, respectively. As expected, the simulated dataset contains several highly correlated features. All calculated coolant parameters, $T_{c,o}$, dP_c, $\dot{m}_c$ and $v_{\%}$, are correlated, as are $\dot{m}_h$ and dP_h for the process fluid. The same correlations are found for real facilities, although with a lower degree of correlation. This indicates that feature selection may be applicable for some correlated features, however in itself does not suffice to conclude correlated features are redundant.

Figures D.4a and D.4b show the correlation difference between training and testing data. These matrices are calculated by subtracting the correlation matrix of the testing data from the correlation matrix of the training data for each facility. Figure D.4a shows three notable correlation differences for the coolant inlet temperature. It can be concluded that this is by chance rather than the result of any important system behavior. Two of these features, $T_{h,i}$ and $\dot{m}_h$, are input parameters to the simulations. The third, dP_h, is inherently correlated with $\dot{m}_h$.

Figure D.4b shows only one noteworthy difference, occurring for the correlation difference between dP_c and $T_{c,i}$. This indicates that there is less correlation between these features when fouling is present than when not. In theory, when coolant inlet temperature increases, more coolant flow is needed to achieve the desired process outlet temperature, and thus the pressure drop for the coolant should increase. Why this relation is less noticeable when fouling is present is not clear. A possible explanation is that other factors which are not necessarily correlated with the coolant inlet temperature contribute to the pressure drop, thus reducing the apparent correlation between these two features.

7.2 Supervised methods

The performance of each regression model is evaluated based on ability to detect fouling. For facility D, this is defined objectively because the fouling level is known. For facilities F and G, empirical evaluations are made, as described in section 4.3.1. Metrics like loss, coefficient of determination and convergence patterns are presented and discussed for relevant models. Each facility is presented separately. Predictive model A is applied first to determine whether the fouling levels can be accurately estimated using the reduced number of sensors, before the remaining predictive models are applied. Unless otherwise is specified, dataset D with fouling

[2] https://github.com/hermanwh/master-thesis/blob/master/1_correlation.ipynb

on the coolant side is used. Note that despite consistently being referred to as the deviation between predicted and measured output value, the graphs shown and discussed are in fact predicted values subtracted from measured values. This order is preferred so that a loss in performance is indicated by a lower output value, and vice versa.

Individual notebooks for each facility are implemented according to the top-level module described in Chapter 5, for facility D with fouling on the coolant side[3], facility D with fouling on the process side[4], facility F[5] and facility G[6], respectively. Additionally, notebooks containing feature comparisons for all facilities are also found on GitHub, for linear[7], MLP[8] and LSTM[9] models, respectively. Furthermore, a separate notebook for benchmark models is implemented[10].

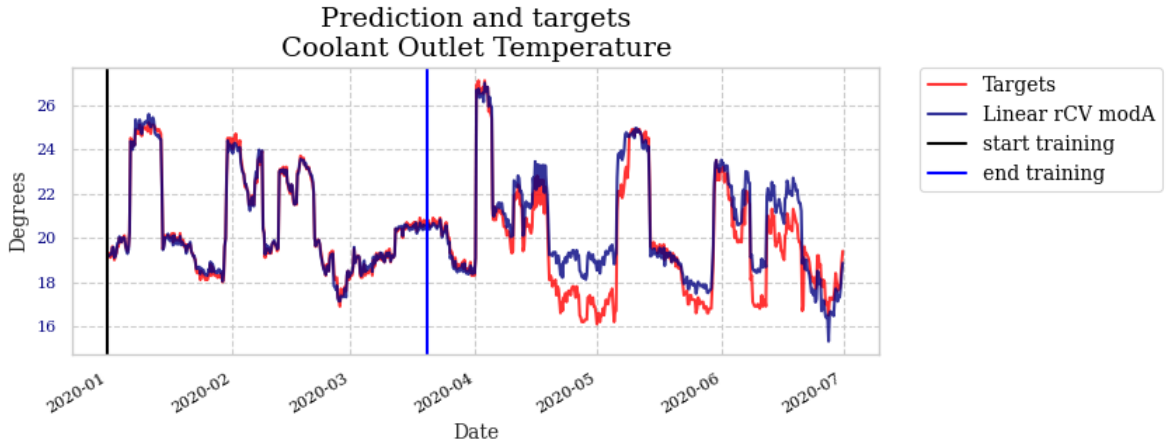
7.2.1 Facility D

Figure 7.2a shows the predicted and measured values for $T_{c,o}$ when using predictive model A with a linear regression model. Figure 7.2b shows the corresponding deviation compared to the added levels of fouling. As fouling increases, a higher outlet temperature is predicted for the coolant than what is measured in practice, indicating that the coolant is absorbing less heat than under ideal circumstances. The deviation pattern fits the added fouling very well, with an apparent linear relationship. During the first linear increase, the estimator is particularly accurate. During the second linear increase, some larger variations can be seen. At the end of the second fouling period, the deviation settles on approximately 0.5 degrees. Despite these problems, use of predictive model A with linear regression model appears to give a reasonable estimate of the fouling level for simulated data.

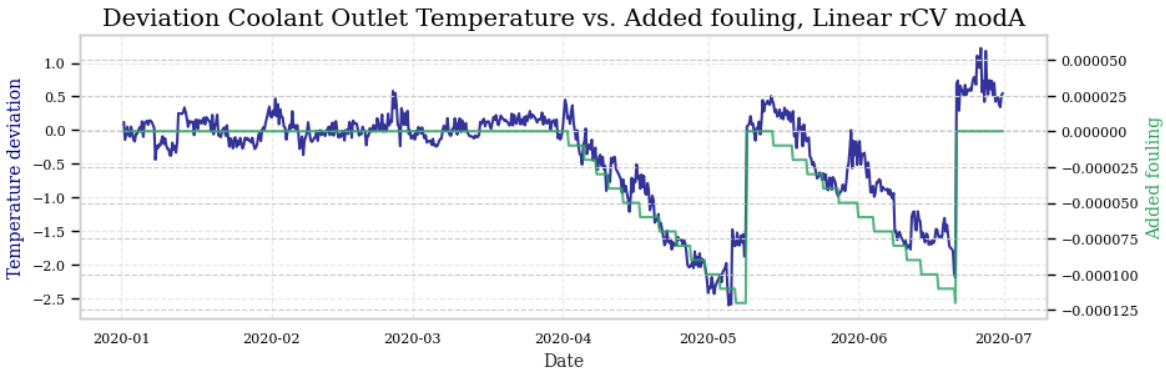
Figure D.5 shows the same graphs for MLP models. A very similar deviation pattern can be seen, with less disturbances during the second linear increase in fouling than for the linear regression model. The value settles closer to zero at the end of the time series, although with some deviation in the positive direction also for this model. As for linear models, predictive model A appears to give a reasonable fouling estimate using MLP models. No significant increases in performance are found through the use of ensemble models with linear and MLP submodels.

LSTM models, regardless of predictive model, are found to perform poorly for dataset D. This is expected, given the very limited amount of samples and large sampling rate for this dataset. Despite not being able to accurately estimate the

[3] https://github.com/hermanwh/master-thesis/blob/master/2_basic_example_D_c.ipynb
[4] https://github.com/hermanwh/master-thesis/blob/master/2_basic_example_D_h.ipynb
[5] https://github.com/hermanwh/master-thesis/blob/master/2_basic_example_F.ipynb
[6] https://github.com/hermanwh/master-thesis/blob/master/2_basic_example_G.ipynb
[7] https://github.com/hermanwh/master-thesis/blob/master/3_model_comp_linear.ipynb
[8] https://github.com/hermanwh/master-thesis/blob/master/3_model_comp_mlp.ipynb
[9] https://github.com/hermanwh/master-thesis/blob/master/3_model_comp_lstm.ipynb
[10] https://github.com/hermanwh/master-thesis/blob/master/3_other_models.ipynb



(a) Predicted and measured values.



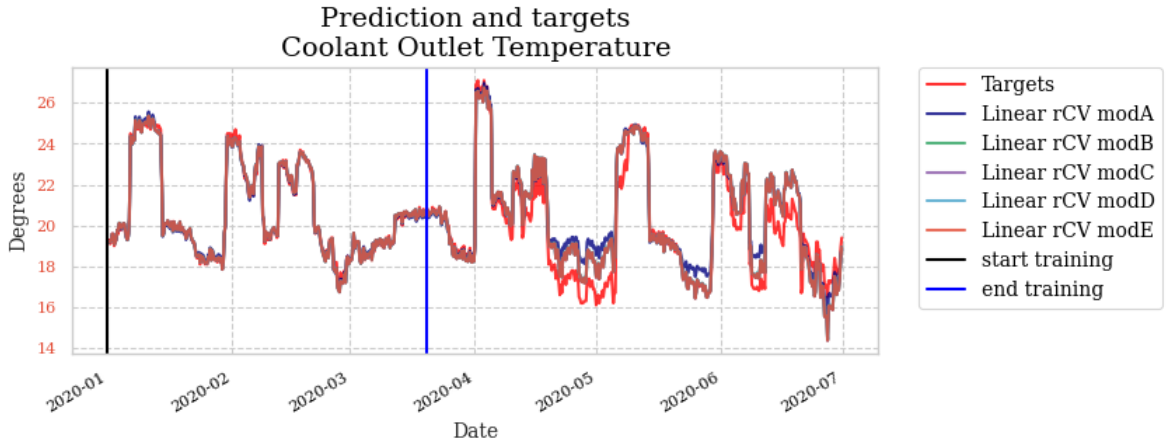
(b) Deviation between predicted and measured values on left axis. Added fouling on right axis. Note that the sign of the added fouling factor has been switched, in order to fit the two graphs.

Figure 7.2: Predictions and deviations for $T_{c,o}$ on dataset D using predictive model A and a linear model with ridge regularization and cross validation.

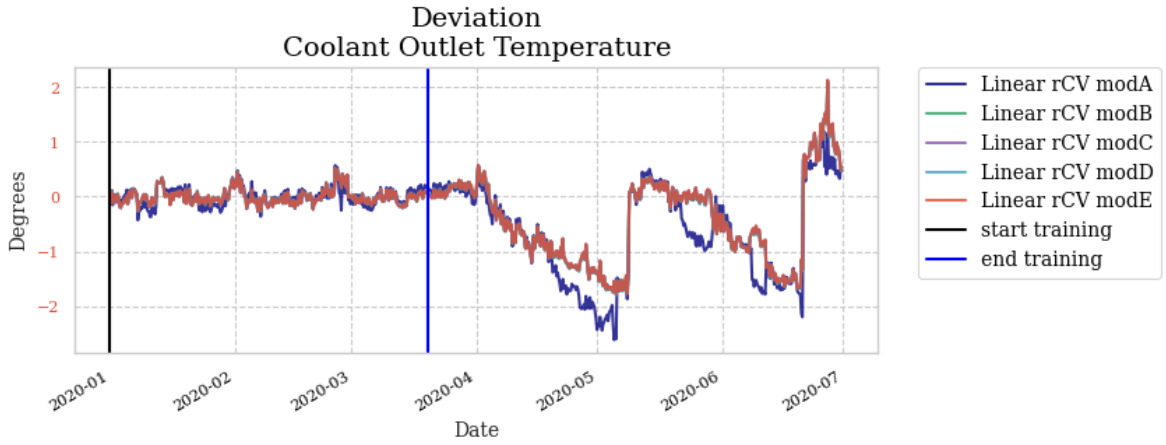
fouling level here, recurrent models are not yet deemed infeasible for modelling heat exchanger data, and will therefore be applied for subsequent datasets.

When additional input parameters are added for predictive models B to E, the linear regression model appears to fit the linear increase in fouling somewhat more accurately. However, this comes at the expense of worse behavior at the end of the time series. The maximum obtained absolute deviation is also slightly lower. Predictions and deviations for these models are seen in Figure 7.3. Almost identical predictions are made for predictive models B to E, indicating that the added features $\dot{m}_c$ and $\nu_{\%}$ are entirely correlated for the simulated data.

Predictions and deviations for MLP models using additional input parameters are seen in Figure D.6. Predictive models B to E appear to be inapplicable. All models show clear signs of overfitting, with acceptable training and validation metrics



(a) Predicted and measured values.



(b) Deviation between predicted and measured values.

Figure 7.3: Predictions and deviations for $T_{c,o}$ on dataset D using predictive models A to E.

despite poor capabilities of estimating the added fouling level. The only notable exception is for predictive model C, although this model also experiences a large deviation from the expected fouling pattern.

The calculated performance metrics are very similar for each of the predictive models, as seen in tables 7.1 and 7.2. The weight for each feature of the linear models are seen in Table 7.3. Every model puts large emphasis on the process flow rate. $T_{h,i}$ is given a sizeable weight, while $T_{h,o}$ and $T_{c,i}$ are given almost zero-value weights. With regards to the dataset, in which these two parameters see little variation, this is sensible. For a dataset with real variations, these parameters are expected to contribute more to the output predictions. Emphasis on $\dot{m}_h$ is maintained for all models even when measurements for $v_{\%}$ and $\dot{m}_c$ are included.

No significant differences between predicted and measured dP_h are found for the simulated dataset with process side fouling, as seen in Figure D.7, compared to the dataset with coolant side fouling. The deviation pattern does not represent anything like the expected fouling pattern. This indicates that deviation between predicted and measured dP_h cannot be used to determine which side of the heat exchanger fouling occurs on for the simulated dataset. This is as expected, due to the lacking hydraulic effects of fouling discussed in section 6.4.6. The inclusion of dP_h as an output parameter is found to affect the model convergence, indicating that more accurate predictive models for $T_{c,o}$ may be obtained by excluding this output parameter.

To conclude, fouling estimation using linear and MLP models is considered very promising, particularly using predictive model A. Performance for recurrent models may not be evaluated for dataset D due to its limited sample size.

Table 7.1: Features comparison R^2, dataset D

Model	Linear	LSTM	MLP
A	0.9947	0.9698	0.9974
B	0.9963	0.9623	0.9972
C	0.9963	0.9761	0.9980
D	0.9963	0.9684	0.9976
E	0.9963	0.9604	0.9754

Table 7.2: Features comparison metrics, dataset D

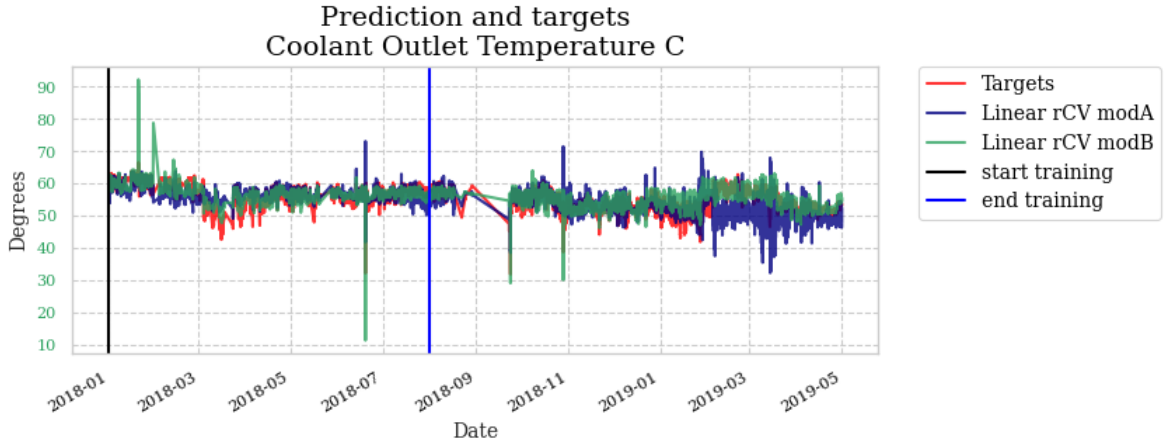
	LSTM			MLP		
Model	Min.loss	Loss	Val.loss	Min.loss	Loss	Val.loss
A	0.0684	0.0872	0.1254	0.0915	0.1099	0.0454
B	0.0765	0.0876	0.1339	0.1028	0.1172	0.0422
C	0.0554	0.0739	0.1242	0.1022	0.1297	0.0340
D	0.0702	0.0923	0.1208	0.0984	0.1179	0.0414
E	0.0771	0.0943	0.1386	0.1259	0.1899	0.0359

Table 7.3: Linear model weights $T_{c,o}$, dataset D

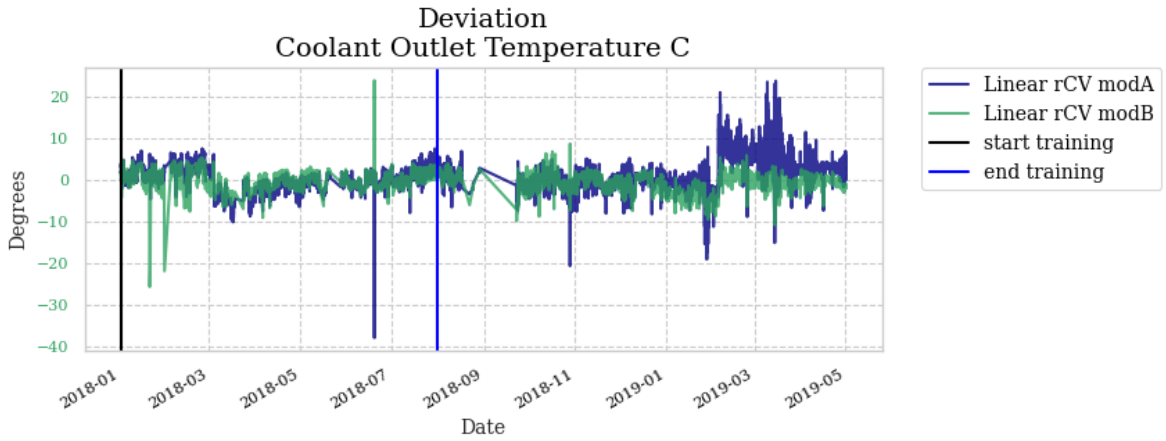
Model	$T_{h,i}$	$T_{h,o}$	$\dot{m}_h$	$T_{c,i}$	$v_{\%}$	$\dot{m}_c$	$P_{c,i}$
A	-0.1458	0.0414	-0.9827	-0.0390	-	-	-
B	-0.0865	0.0377	-0.8175	-0.0242	-0.1816	-	-
C	-0.0856	0.0380	-0.8131	-0.0241	-	-0.1859	-
D	-0.0865	0.0377	-0.8175	-0.0242	-0.1816	-	0.0000
E	-0.0856	0.0379	-0.8133	-0.0240	-0.0294	-0.1563	0.0000

7.2.2 Facility F

Predictions and deviations for predictive models A and B using linear regression models are seen in figures 7.4a and 7.4b, respectively. A considerable increase in R^2 metric is observed for predictive model B, as seen in Table 7.5, despite two very noticeable outlier predictions around the start of the time series for this model.



(a) Predicted and measured values.



(b) Deviation between predicted and measured values.

Figure 7.4: Predictions and deviations for $T_{c,o}$ on dataset F using a linear model with ridge regularization and cross validation.

Model A shows extensive variation in its predictions compared to model B, particularly following the heat exchanger cleaning just before the timestamp 2019.03 in the figure. Additionally, predictions seem to be gradually drifting towards lower predicted outlet temperatures. This causes a large positive deviation between predicted and measured value following the heat exchanger cleaning. While slightly positive deviation is expected following cleaning due to limited fouling in the fa-

cility during the training data, deviations as large as 10 degrees is considered unrealistic.

Meanwhile, the predictions of predictive model B appear more logical. A clear jump back to a slightly positive deviation is seen following the heat exchanger cleaning, as well as a somewhat linearly increasing pattern before and after. These are the same properties for the fouling pattern found previously for facility D. However, these increases are in the same order of magnitude as the errors calculated on the training data. Thus, it is not possible to pinpoint these increases to increased fouling, despite some strong indications.

Table 7.4: Linear model weights $T_{c,o}$, dataset F

Model	$T_{h,i}$	$T_{h,o}$	$\dot{m}_h$	$T_{c,i}$	$v_{\%}$	$\dot{m}_c$	$P_{c,i}$
A	0.4507	0.3595	-0.2272	-0.0655			
B	0.3586	0.1779	0.5177	0.0089	-0.9775		

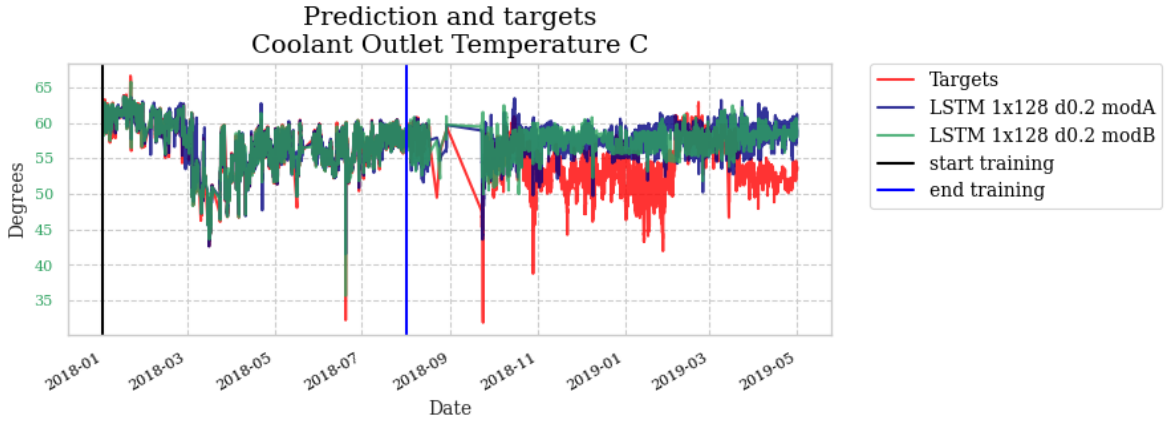
From the linear model weights seen in Table 7.4, an interesting observation is made. The inclusion of the $v_{\%}$ parameter causes the $\dot{m}_h$ weight to change value considerably, from -0.2272 to 0.5177. A negative weight for $\dot{m}_h$ indicates that coolant outlet temperature decreases when process flow rate increases. This aligns with heat transfer theory presented previously, as more coolant flow is required to cool the increased process flow, and thus each mol of coolant will receive less heat. A negative weight for $v_{\%}$ indicates the same, because a higher valve opening generally results in higher coolant flow. Therefore, the positive weight obtained for $\dot{m}_h$ using predictive model B is not supported by heat transfer theory. Hence, the model appears to have overfitted to a combination of $\dot{m}_h$ and $v_{\%}$ weights which score well for the training data and perform acceptably for the testing data, however with some illogical weights when theoretical considerations are made.

MLP models for neither predictive model achieve desirable performance. As seen in Figure D.8, the predictions show much of the same variations as discussed previously for the linear model with predictive model A. The deviation jumps considerably following the heat exchanger cleaning, although to a rather high positive deviation of approximately 5 degrees. A slight average decrease in deviation is seen, but once more this is in the same order of magnitude as the deviation during the training data. Calculated loss metrics are considerably worse than LSTM models, and only slightly better than linear models, as seen in Table 7.6. R^2 score is slightly higher than for the linear model. Overall, MLP models appear unsuitable for dataset F given the applicable predictive models A and B.

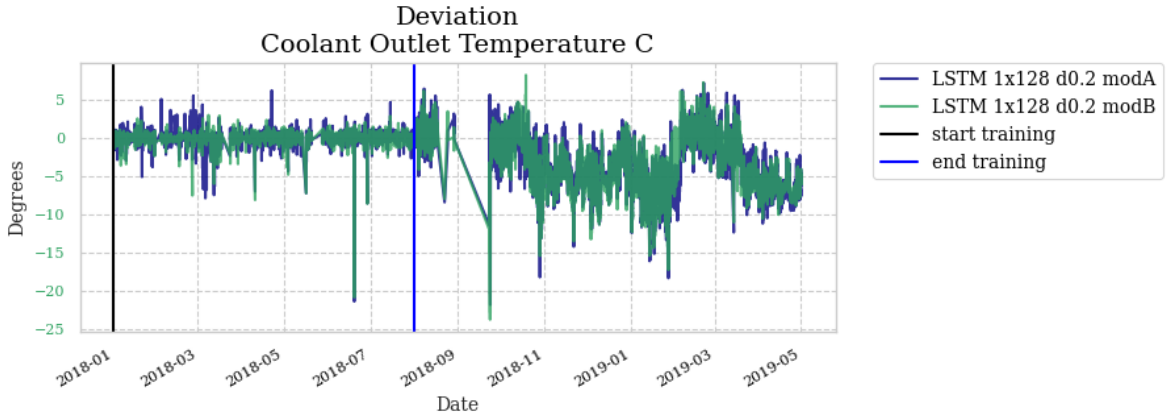
LSTM models show much more promise when applied for dataset F, for both predictive models A and B. Figures 7.5a and 7.5b show the predictions and deviations for both models, respectively. Small errors with considerably less variation than for linear and MLP models are observed during the training period. Predictions during the testing period vary around a relatively constant mean. The deviation

pattern shows two somewhat linear increases in average predictions, with a noticeable jump back to a small deviation following the heat exchanger cleaning. Unlike for the previous models, the positive deviation following the cleaning is very small. Although the peaks of the variation around this point reach a positive deviation of approximate 5 degrees, the mean is around 1.5 degrees, which is considered more reasonable given some fouling in the training data.

As for the linear models, the increases in deviation cannot be pinpointed to occur as a result of fouling because no known instances of fouling are known for the facility. However, the similarities between the deviation patterns for datasets F and D strongly suggests fouling is present in the heat exchanger, causing a maximum deviation between predicted and measured $T_{c,o}$ of approximately 6 degrees. This is roughly three times as much as for dataset D, indicating a fouling factor of $36 \cdot 10^{-5} m^2 K/W$.



(a) Predicted and measured values.



(b) Deviation between predicted and measured values.

Figure 7.5: Predictions and deviations for $T_{c,o}$ on dataset F using LSTM models.

Table 7.5: Features comparison R^2, dataset F

Model	Linear	LSTM	MLP
A	0.3234	0.9606	0.4342
B	0.6158	0.9800	0.7187

Table 7.6: Features comparison metrics, dataset F

	LSTM			MLP		
Model	Min.loss	Loss	Val.loss	Min.loss	Loss	Val.loss
A	0.0521	0.0989	0.1131	0.4930	0.5217	0.5546
B	0.0256	0.0452	0.1378	0.3146	0.3862	0.4862

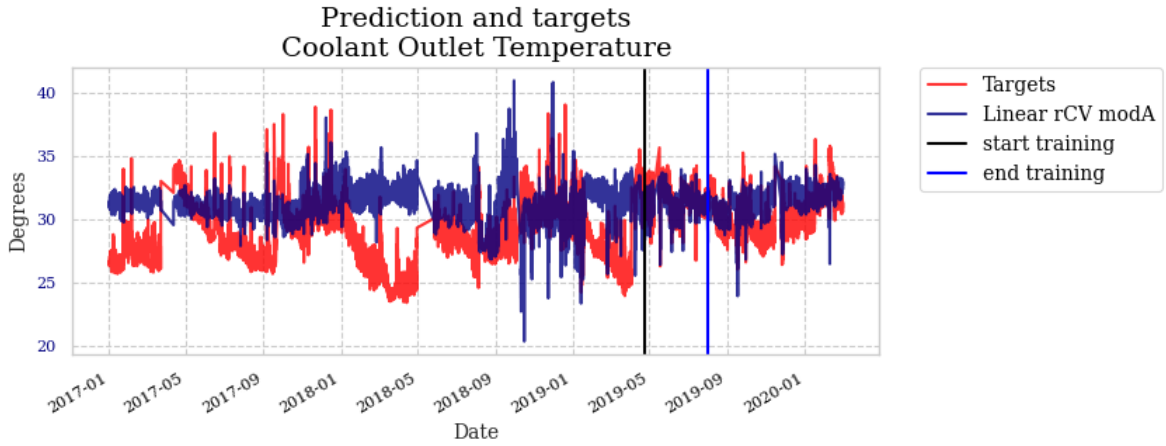
7.2.3 Facility G

Unlike facility F, facility G has experienced continued problems related to fouling. Recall the maintenance dates listed in Table 6.4. If similar behavior using the proposed fouling indicators is found in relation to each documented maintenance, this may serve as further proof that the predictive models are applicable.

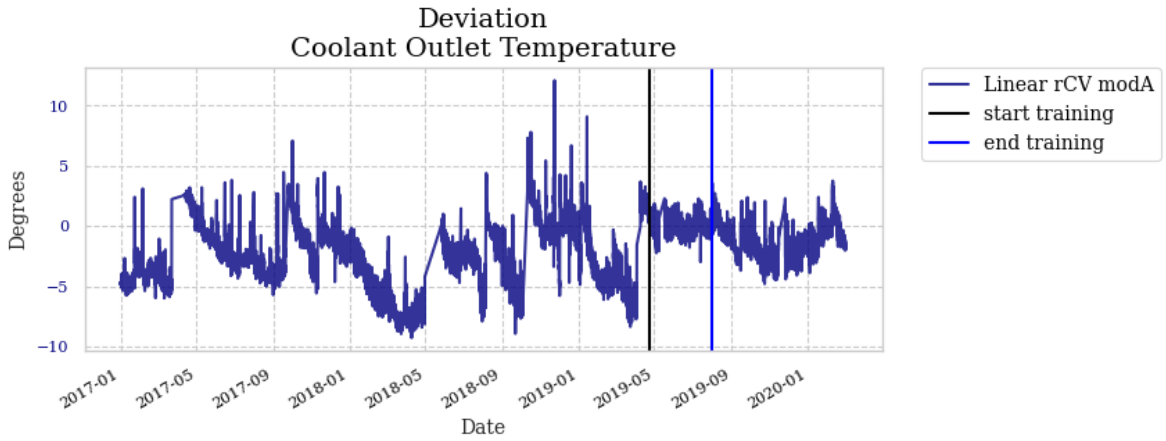
Figure 7.6 shows predictions and deviations for model A using a linear regression model. The predictions in general seem reasonable, although with some large variations around January 2019. Clear leaps back towards zero deviation are visible in relation to each of the indicated maintenance dates, with the exception of the acid wash in January 2018. A similar leap is seen around November 2017, for which no maintenance is registered. The deviation often does not reset back to zero, e.g. reaching a deviation of approximately 2 degrees following March and September 2019, and approximately -2 degrees following May 2018. These differences are somewhat sensible considering that the operating conditions may have changed slightly throughout the three year long span of the prediction interval. The maximum deviation of approximately -7 degrees is in the same order of magnitude as was discovered for facility F. Overall, considering the maintenance history of the facility, the predictive model appears to indicate the level of fouling well.

Table 7.7: Features comparison R^2, dataset G

Model	Linear	LSTM	MLP
A	0.5620	0.8797	0.6775
B	0.9235	0.8588	0.9454
C	0.9022	0.8920	0.9100
D	0.9239	0.8683	0.9534
E	0.9238	0.8438	0.9462



(a) Predicted and measured values.



(b) Deviation between predicted and measured values.

Figure 7.6: Predictions and deviations $T_{c,o}$ on dataset G using linear model A.

R^2 score for each predictive model and regression model are listed in Table 7.7. Notice that both linear and MLP models see a drastic increase in metric value from model A to model B, with rather persistent high performance for models B through E. Unlike what was discovered for dataset F, the score of linear and MLP models for these predictive models is considerably higher than for LSTM models. It becomes apparent by looking at the prediction and deviation plots for predictive models B, D and E that the trained linear and MLP models for these are not valid solutions. Figure D.9 shows how the linear model overfits to the $v_{\%}$ parameters when using predictive model B. Plot for the MLP model using the same predictive model can be seen in Figure D.10. Similar overfitting is experienced for predictive models D and E. This relates to the problematic valve opening behavior discussed in section 6.4.4. As seen for the linear weights in Table 7.9, the weight of the $v_{\%}$ parameter is much higher than for the remaining weights for models B, D and E.

A similar weight relation can be seen for predictive model C, for which $\dot{m}_c$ receives a particularly large weight. However, this model does not overfit in the same manner as the previously mentioned models. Predictions and deviations for the linear model are shown in Figure D.11. Although some of the same behavior as for predictive model A can be seen, e.g. following specific maintenance dates, the results in general are more difficult to interpret. For instance, a linear decrease in predicted $T_{c,o}$ can be observed follow the first maintenance from around May to September 2017. This decrease in itself is not illogical considering the coolant flow rate is increasing throughout the same period, making each mol of coolant absorb less heat. However, the predictions closely follow the measured values, resulting in a rather small deviation of approximately −2 degrees. This is considerably less than the deviation found for predictive model A and for facility F of −6 degrees, however very similar to the maximum deviation observed for facility D. Which model is more accurate is difficult to determine without known fouling conditions for the heat exchanger.

The difficulties of accurately fitting linear and MLP models when including the valve opening indicate that the use of this parameter is challenging when fouling problems are experienced, because the flow rate does not change accordingly when the valve opening is regulated at high $v_{\%}$ values. For facility F, the use of this parameter did not pose the same problems because the valve opening in general behaves normally during the time span of the acquired data. For facility D, valve opening was entirely correlated with the coolant flow rate. However, for facilities F, the valve opening is regulated drastically, as seen in Figure 6.4b.

Table 7.8: Features comparison metrics, dataset G

	LSTM			MLP		
Model	Min.loss	Loss	Val.loss	Min.loss	Loss	Val.loss
A	0.1408	0.2145	0.2298	0.4004	0.4164	0.4932
B	0.1492	0.2188	0.2957	0.1660	0.1805	0.1967
C	0.1173	0.1765	0.2765	0.1937	0.2216	0.3148
D	0.1380	0.2104	0.2784	0.1590	0.1623	0.1722
E	0.1605	0.2340	0.3046	0.1580	0.1732	0.2016

Table 7.9: Linear model weights $T_{c,o}$, dataset G

Model	$T_{h,i}$	$T_{h,o}$	$\dot{m}_h$	$T_{c,i}$	$v_{\%}$	$\dot{m}_c$	$P_{c,i}$
A	-0.1804	0.2611	-0.7434	-0.2685			
B	0.0665	0.2810	0.0151	-0.0040	-0.9425		
C	0.0322	0.2407	0.0410	-0.1551		-0.9511	
D	0.0663	0.2818	0.0158	-0.0038	-0.9434		-0.0206
E	0.0647	0.2782	0.0226	-0.0089	-0.8634	-0.0863	-0.0194

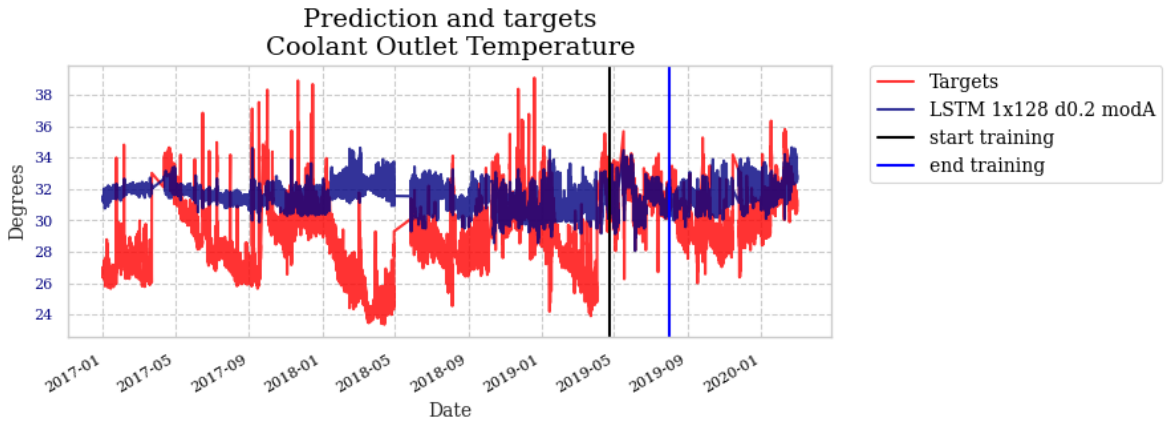
LSTM models once more show a lot of promise. Unlike linear and MLP models, the predictions performed using predictive models A, B and C are very similar. No notable differences are observed when adding additional parameters for predictive model D and E. Predicted and measured values can be seen in Figure 7.7. The corresponding deviation graphs are shown in Figure 7.8. Predictions for predictive model A are quite similar to those of the linear model, although with less variations in general and without the large variations noted around January 2019. Hence, the deviation pattern is also similar, with notable jumps occurring after each maintenance.

Predictions for predictive models B and C show some of the same behavior as discussed for predictive model C using linear models, with slight decreases in $T_{c,o}$ prediction between each maintenance. However, unlike for the linear model, there is still a clear and increasing deviation between the predicted and measured value for each period. This indicates that the coolant outlet temperature is decreasing because coolant flow is increasing, which is captured by the LSTM model. Simultaneously, fouling is increasing in the heat exchanger, causing further decreases in coolant outlet temperature which are not captured by the LSTM model. The resulting deviation appears to indicate an appropriate fouling level.

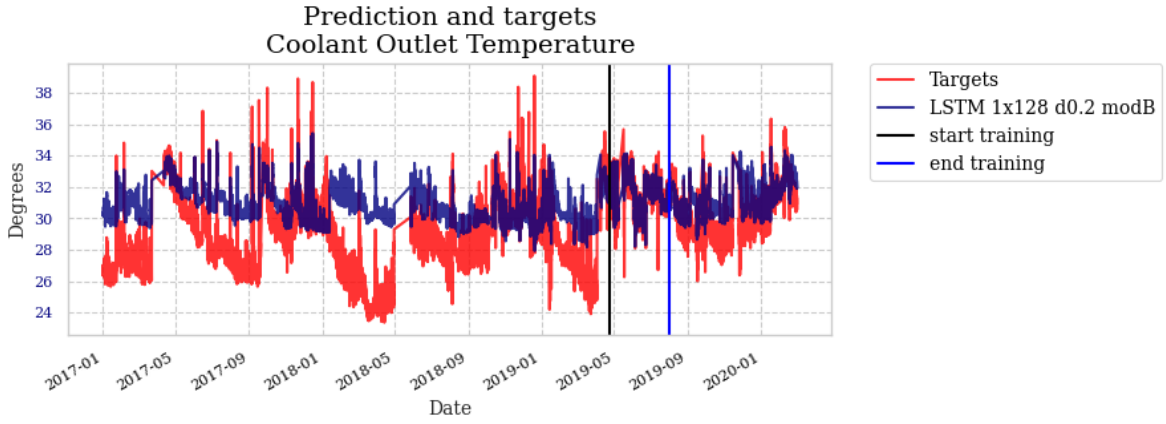
Some significant differences from model A and C can be seen for model B around January 2018. This corresponds to the portion of data for which it was noted in section 6.4.4 that valve opening and coolant flow are not necessarily correlated. This suggests that although there are no strong indications of overfitting for the LSTM model using predictive model B, it still experiences some issues when predicting with large valve opening values.

Although neglected in favor of $T_{c,o}$ predictions after findings for facility D, individual models are trained with dP_h as output parameter to determine if deviations between predicted and measured process side pressure difference can give indications as to which side of the heat exchanger fouling is present. Predictions and deviations for a linear model using predictive model A are shown in Figure D.12. These results are not considered suitable as a fouling indicator. For process side fouling, the predicted pressure drop should remain stable while the measured value is increasing, according to **hypothesis 2** proposed in section 4.2.5. Similarly for coolant side fouling, the predicted and measured process side pressure drop should remain stable. While some stable areas of deviation are seen in Figure D.12b, there are also periods of decreasing deviation which cannot logically be explained according to the thesis methodology. This in itself does not exclude pressure drop as a possible fouling indicator, however it appears difficult to draw any conclusions with the limited emphasis placed on such methods in this thesis.

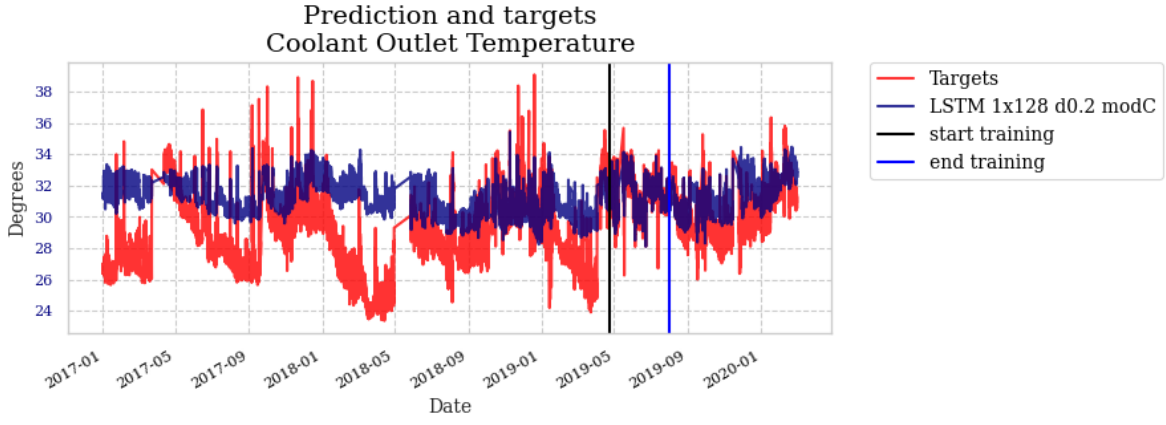
Overall, the use of LSTM models, and predictive models A and C with only $T_{c,o}$ as output parameter in particular, are considered very encouraging as fouling indicators. Linear models also show some promise, although prone to overfitting. MLP models provide no clear benefits over simpler, linear models.



(a) Predicted and measured values, predictive model A.

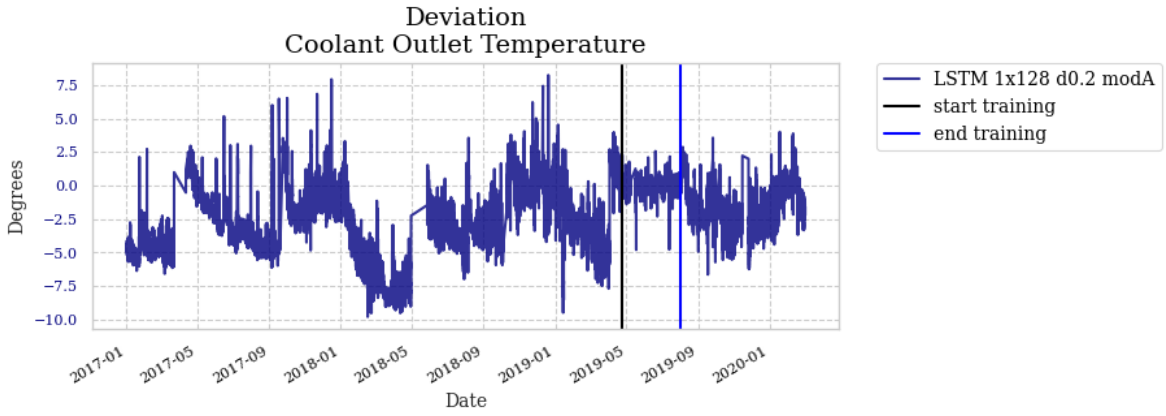


(b) Predicted and measured values, predictive model B.

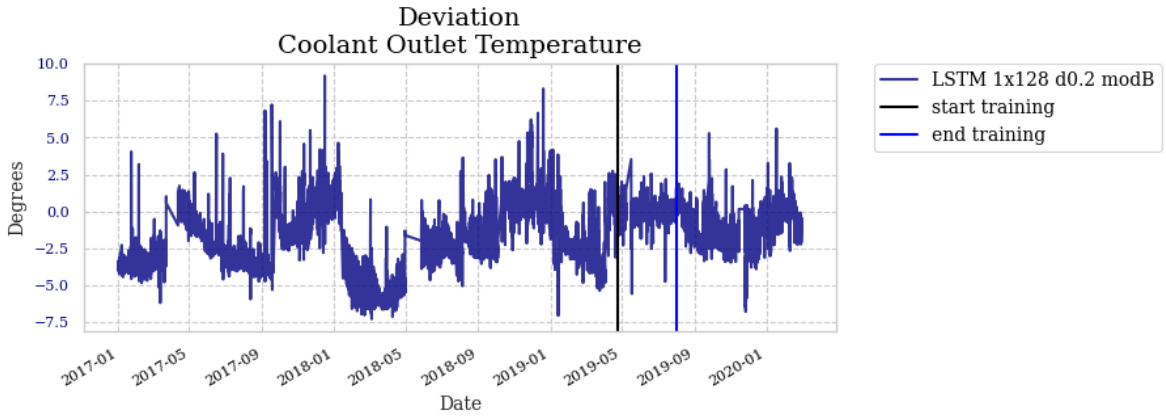


(c) Predicted and measured values, predictive model C.

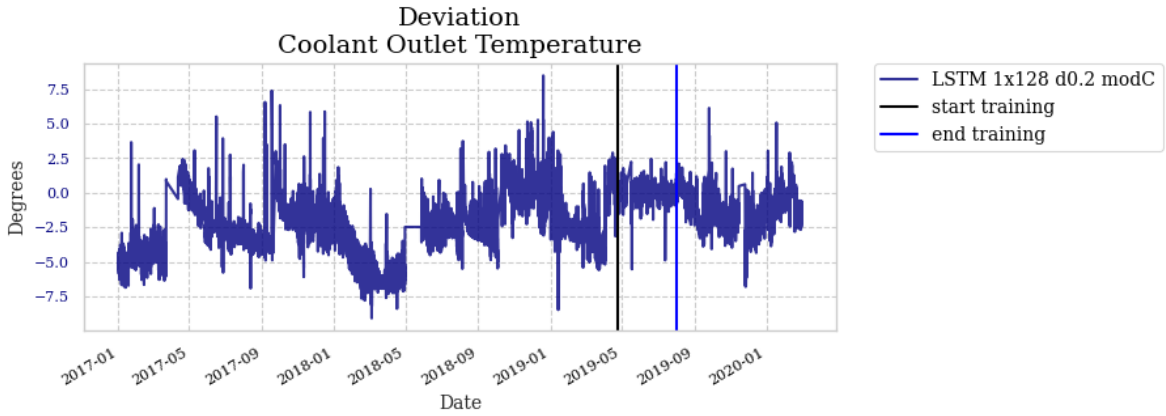
Figure 7.7: Predictions for $T_{c,o}$ on dataset G using LSTM models A-C.



(a) Deviation between predicted and measured values, predictive model A.



(b) Deviation between predicted and measured values, predictive model B.



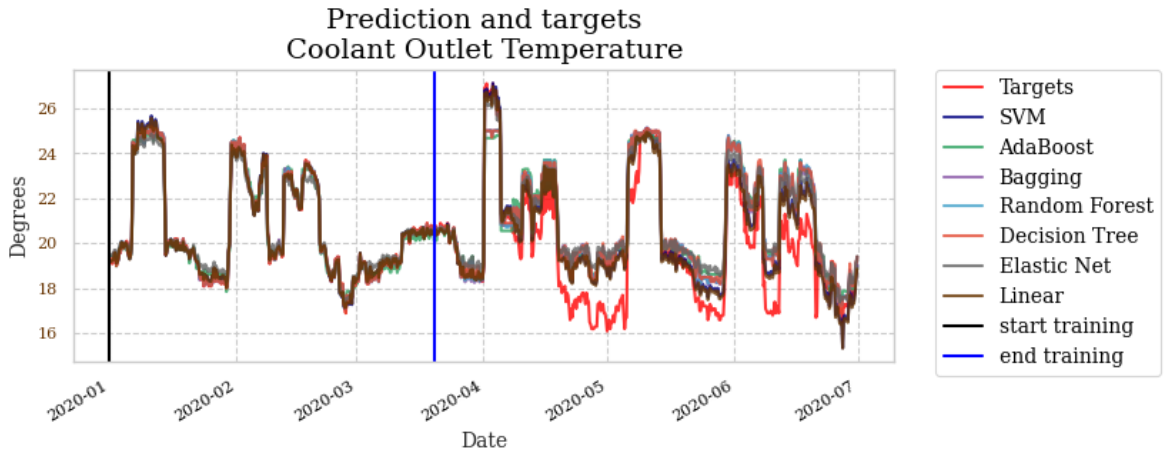
(c) Deviation between predicted and measured values, predictive model C.

Figure 7.8: Deviations between predicted and measured values for $T_{c,o}$ on dataset G using LSTM models.

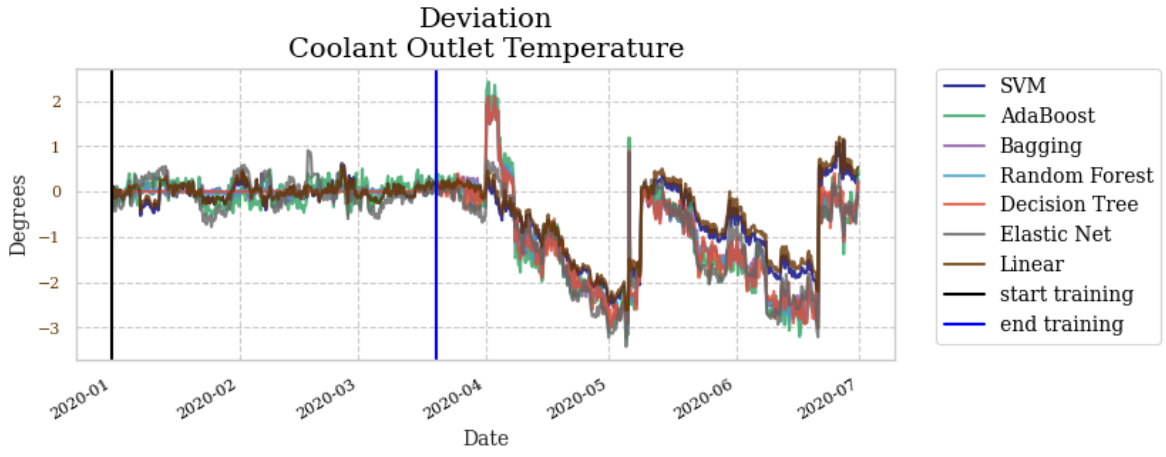
7.2.4 Benchmarks

Additional model benchmarks

Regression models available in the Scikit-Learn library are implemented as benchmark models. As noted in section 4.3.5, limited time is put into understanding and tweaking the benchmark models. Hence, their validation is questionable, and increased performance could most certainly have been obtained through tweaking the model parameters. Predictions and deviations using predictive model A for dataset D are seen in Figure 7.9. Calculated metrics for all datasets are seen in Table 7.10.



(a) Predicted and measured values.



(b) Deviation between predicted and measured values.

Figure 7.9: Predictions and deviations for $T_{c,o}$ on dataset D using benchmark models.

Tree-based models such as decision tree, random forest and bagging regression models show clear signs of overfitting. Deviation during the testing period is near zero, and for the decision tree model the deviation is in fact zero. These models follow the added fouling pattern reasonably well, although with some large deviations, e.g. at the beginning of the testing period. The degree of overfitting indicates that tree-based models are not applicable as fouling indicators. The remaining models, namely support vector regression, AdaBoost and elastic net, perform very similarly to the linear model. Some notable differences are observed, most importantly a large spike in predictions just after timestamp 2020.05. The AdaBoost regressor shows the same spike as the tree-based methods around timestamp 2020.04. Additionally, the predicted deviation is slightly higher for several models, while the final predicted deviation level deviates from that of the linear model.

The same patterns as observed for dataset D are found for datasets F and G. The tree-based methods overfit and achieve very high R^2 metrics, while the remaining models predict and perform very similarly to the linear model. Based on these limited findings for predictive model A, some of the benchmark models may be equally applicable as fouling indicators as linear regression models. However, considering the limited model explainability of such models compared to linear models, it is debatable whether further research for these is worthwhile, especially given the success of LSTM models.

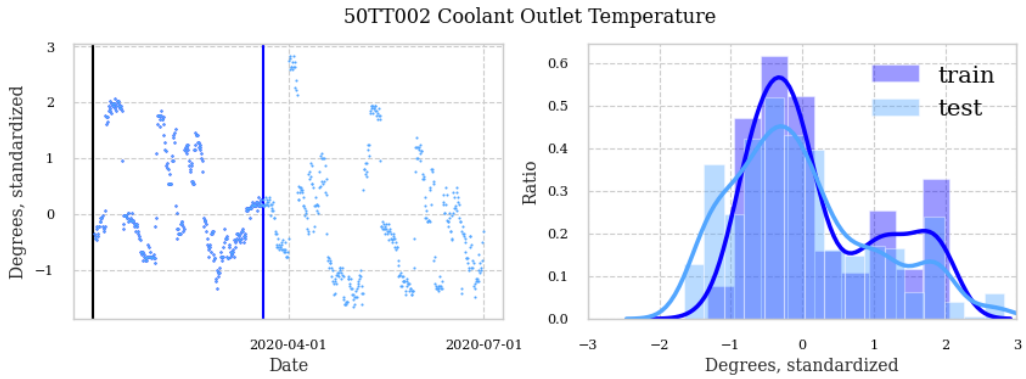
Table 7.10: Additional model metrics, datasets D, F and G

Model	D	F	G
Linear	0.9947	0.3234	0.5620
SVM	0.9945	0.3072	0.5543
AdaBoost	0.9881	0.3838	0.5664
Elastic Net	0.9797	0.3053	0.5378
Bagging	0.9989	0.9387	0.9513
Random Forest	0.9993	0.9554	0.9672
Decision Tree	1.0000	1.0000	1.0000

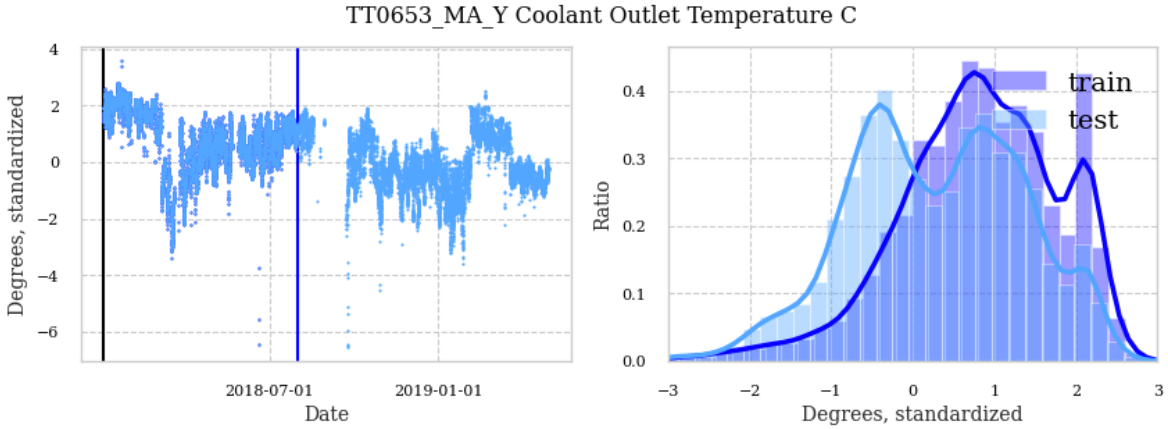
Manual parameter inspection

The second benchmark strategy is manual inspection of dataset parameters, most importantly the coolant outlet temperature. Plots of this parameter for facilities D, F and G can be seen in figures 7.10a, 7.10b and 6.2, respectively.

A significant challenge when applying manual parameter inspection is that there are no available varying reference levels for the expected behavior, only static design values or historical means. While reasoning whether a value should be increasing or decreasing may be possible using heat exchanger theory, it is difficult to determine how large increases or decreases are deemed appropriate.



(a) Coolant outlet temperature for dataset D.



(b) Coolant outlet temperature for dataset F.

Figure 7.10: Coolant outlet temperature for datasets D and F.

The coolant outlet temperature for facility D varies according to the drastic changes in simulation input parameters, as discussed in section 6.4.5. Although the temperature should decrease when fouling increases, it is not possible to identify such a relation using the plots, because values are consistently changing between predefined levels. In a more realistic facility, parameters would remain at similar levels for an extended period of time, enabling manual inspection to a larger extent. Even though the variations are somewhat unrealistic for a real processing facility, the difficulty of determining fouling levels using plots in this case highlights the need for predictive models rather than relying on reference values and manual inspection.

For facility F, a clear decrease in coolant outlet temperature leading up to the heat exchanger cleaning is observed, followed by a leap towards higher temperatures afterwards. Because the figure axis uses standardized values, the extent of the decrease and jump in terms of degrees can be derived by using the standardized values and the dataset statistics in Table E.2. A difference of approximately 7 de-

grees following the cleaning is found. Operating conditions otherwise are rather similar following the cleaning, indicating that this difference in $T_{c,o}$ may be attributed to fouling. However, the main purpose of condition monitoring is not to derive the level of fouling after the fact. Without knowledge of the temperature following the cleaning, it is difficult to evaluate how much of the decrease in temperature can be ascribed to fouling. For a large period of the time series, the facilities sees increases in process flow rate. To maintain the desired process outlet temperature, it is natural to assume that the coolant flow rate has increased analogously, with a corresponding reduction in coolant outlet temperature. This makes manual inspection challenging to use for this facility as well, at least as a mean of determining fouling levels ahead of maintenance rather than afterward.

For facility G, similar patterns of decreasing coolant outlet temperature can be seen, with leaps following each cleaning. This time, the process flow rate of the facility is relatively stable, and thus uncertainty in the fouling level related to this parameter is substantially lower than what was the case for facility F. However, a different factor which affects $T_{c,o}$ is present, namely the seasonal variation in $T_{c,i}$ due to the use of seawater coolant. Note also that the coolant outlet temperature in general varies throughout the time series, and that the temperature levels following each maintenance are not equivalent. While it is certainly possible to use manual parameter inspection to identify heat exchanger cleanings and potentially estimate fouling levels for this facility, the lack of reference levels makes this challenging in practice. Rough estimates find temperature differences of 6.6, 4.2 and 5.5 degrees following each of the three first cleanings, respectively. If these differences are adjusted according to corresponding coolant inlet temperatures and coolant flow rates, realistic estimates for the temperature differences attributed to fouling may be obtained. Without extensive domain knowledge, this appears infeasible as an accurate fouling estimator. However, engineers operating the facilities in practice may find success using this method.

7.2.5 Fouling estimates

From Figure 7.2, a fouling factor of $12 \cdot 10^{-5} m^2 K/W$ is found to correspond to a deviation in $T_{c,o}$ of approximately -2 degrees. This gives the following expression for the fouling factor R_f as a function of prediction $\hat{T}_{c,o}$ and measured value $T_{c,o}$:

$$R_f(T_{c,o}, \hat{T}_{c,o}) = -(T_{c,o} - \hat{T}_{c,o}) \cdot 6 \cdot 10^{-5}\, m^2/W \tag{7.1}$$

Applying this equation for the maximum deviations found for facility F and G yields fouling factors of approximately $36 \cdot 10^{-5} m^2 K/W$ and $42 \cdot 10^{-5} m^2 K/W$, respectively. Several heat exchanger cleanings for facility G are performed at a deviation of approximately -5 degrees, corresponding to a fouling factor of $30 \cdot 10^{-5} m^2 K/W$. The accuracy of generalizing Eq. 7.1 for arbitrary processing facilities cannot be evaluated without further study of how different operating conditions affect the relationship between calculated deviation and added fouling.

As seen previously for facilities F and G, a positive deviation is often observed after cleaning, indicating higher performance than during the training period. This is caused by small amounts of fouling in the training data. To account for this, a correction factor $T_{f,i}$ may be found and used based on the deviation following heat exchanger maintenance for an arbitrary facility i. This results in the following expression for the fouling factor $R_{f,i}$ for facility i:

$$R_{f,i}(T_{c,o}, \hat{T}_{c,o}) = -(T_{c,o} - \hat{T}_{c,o} - T_{f,i}) \cdot 6 \cdot 10^{-5}\, m^2/W \tag{7.2}$$

Using facility F as an example, for which a deviation of approximately 1.5 degrees was found following maintenance, the correction factor $T_{f,F}$ may be introduced as follows:

$$R_{f,F}(T_{c,o}, \hat{T}_{c,o}) = -(T_{c,o} - \hat{T}_{c,o} - T_{f,F}) \cdot 6 \cdot 10^{-5}\, m^2/W \tag{7.3}$$

$$R_{f,F}(T_{c,o}, \hat{T}_{c,o}) = -(T_{c,o} - \hat{T}_{c,o} - 1.5K) \cdot 6 \cdot 10^{-5}\, m^2/W \tag{7.4}$$

The correction factor $T_{f,i}$ is an indication of how much fouling is present in the training data for facility i. As such, more accurate fouling estimates may be obtained using Eq. 7.2 than Eq. 7.1, because the latter assumes fouling is zero through the entirety of the training period. Note, however, that some uncertainty is introduced by the correction factor. The effects of maintenance are not necessarily equal for each heat exchanger cleaning. Additionally, the fouling level throughout the training period is in fact not constant, but likely continuously increasing. It is unclear how trained models are affected by small amounts of fouling in the training data.

7.3 Predicting across facilities

Determining whether machine learning models trained on a facility can be used to make predictions for other facilities is of great interest. If so, the need to gather training data for new facilities may be eliminated, enabling the use of predictive maintenance from the get-go. To decide whether models used in this manner possess any predictive capabilities, results are presented where models trained for each facility are used to perform predictions for the remaining facilities. Determining whether models possess any meaningful predictive capabilities is difficult in itself, but particularly so when used across facilities. Thorough analysis of both the underlying data sources may be required to understand the variations and differences in predictions, and even then the model performance may not be explainable. Examining the model parameters is out of the question due to the amount of trainable weights, except for linear models. Additional results and plots for cross facility predictions can be seen on GitHub[11].

[11] https://github.com/hermanwh/master-thesis/blob/master/4_pred_cross_mod.ipynb

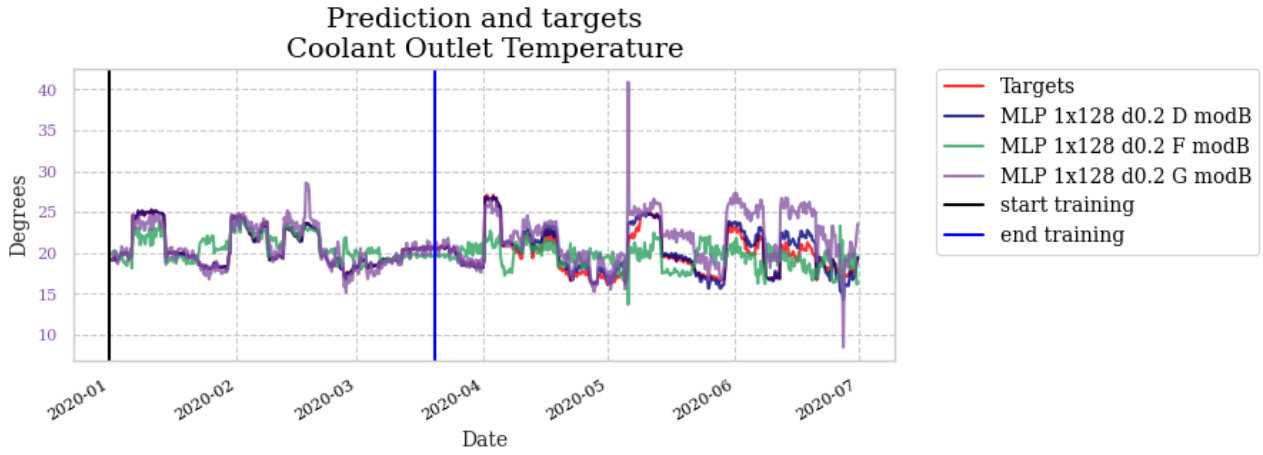
Table 7.11: Cross facility prediction metrics, datasets D, G and G

Dataset	Model	Trained on	Linear	MLP	LSTM
D	A	D	0.9947	0.9982	0.9872
D	A	F	−0.0416	−0.8788	−0.5111
D	A	G	0.8474	0.6001	−0.2707
D	B	D	0.9963	0.9989	0.9795
D	B	F	0.5518	0.4937	0.1004
D	B	G	0.8819	0.8182	0.6743
F	A	D	−0.7919	−0.7766	−0.4382
F	A	F	0.3234	0.5823	0.9680
F	A	G	−0.6057	−0.4216	−0.4221
F	B	D	−0.5317	−0.0405	−0.1382
F	B	F	0.6158	0.8315	0.9844
F	B	G	0.1263	0.0674	0.0294
G	A	D	0.3658	0.3808	0.1138
G	A	F	0.0460	−0.4071	−1.2140
G	A	G	0.5620	0.6984	0.8744
G	B	D	0.5178	0.7469	0.3163
G	B	F	0.6535	0.1446	0.0990
G	B	G	0.9235	0.9540	0.8340

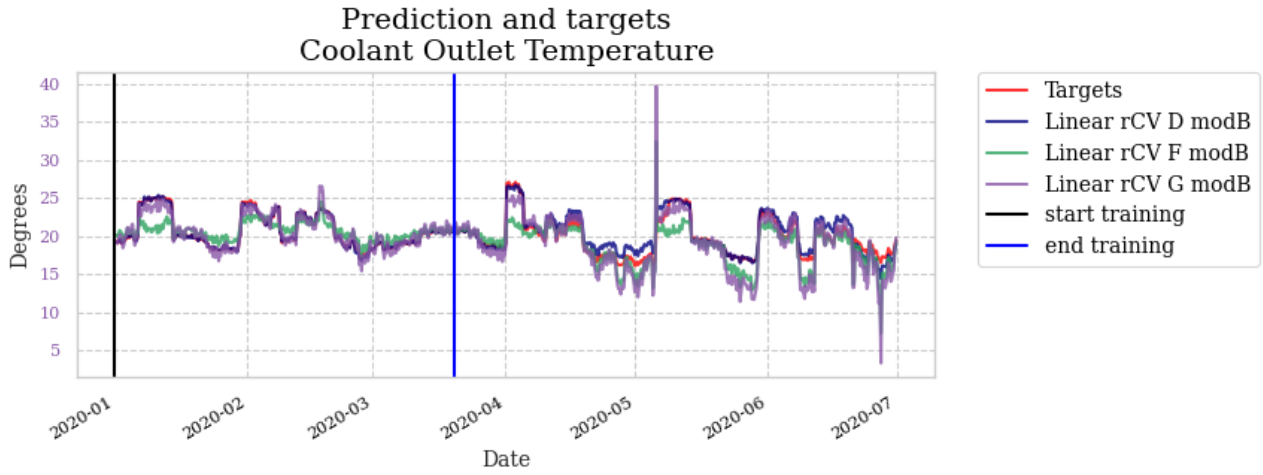
Table 7.11 shows the coefficient of determination for the training data across all facilities for predictive model A and B as defined in section 4.2.5. A single architecture for LSTM and MLP models is used in all cases, with one hidden layer of 128 neurons. Some combinations of predicted dataset, trained dataset and predictive model of particular interest are discussed below.

Models trained on other datasets are shown not to generalize well for facility D, neither using predictive model A nor B. Although the R^2 score calculated on the training set is acceptable at times, it becomes apparent by looking at the plots that the predictions do not follow the expected fouling pattern. Recurrent models were previously established to perform poorly for dataset D in general. The MLP trained on dataset G seems to follow the general trends, however with substantial errors as seen in Figure 7.11a, with two outliers in particular around timestamps 2020.05 and 2020.07. The linear models seen in Figures 7.11b and 7.11c perform similarly, capturing the general trends but with too substantial error to produce a deviation pattern similar to the added levels of fouling.

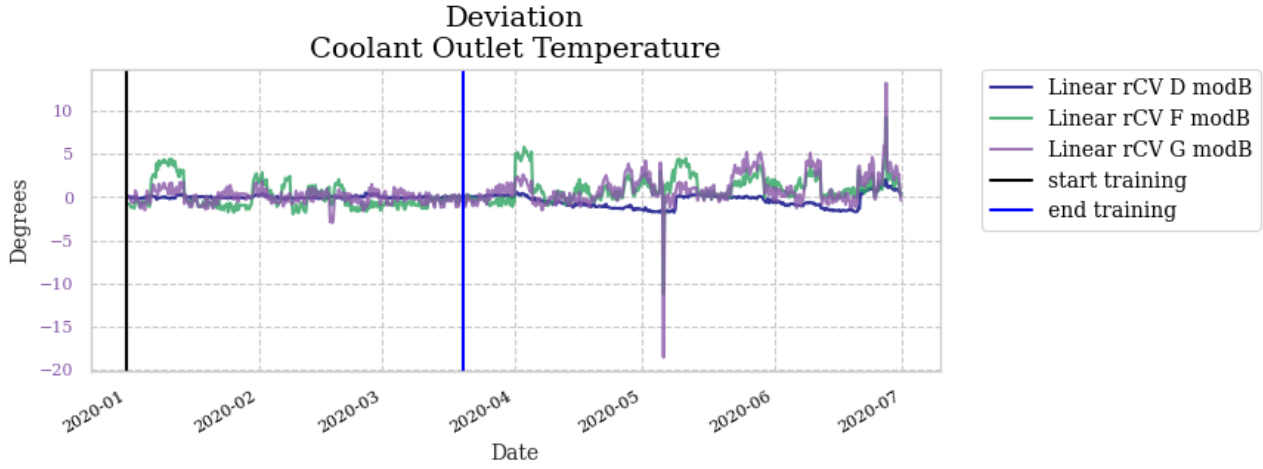
For facility F, neither predictive model A nor B for other datasets show much promising behavior. Predictions for LSTM, MLP and linear models using predictive model B can be seen in figures D.13a, D.13b and D.13c, respectively. The MLP model trained on dataset G using predictive model A shows particularly large deviations, often predicting output values wrong by several orders of magnitude,



(a) Predictions for facility D using MLP models trained on facilities D, F and G.



(b) Predictions for facility D using linear models trained on facilities D, F and G.

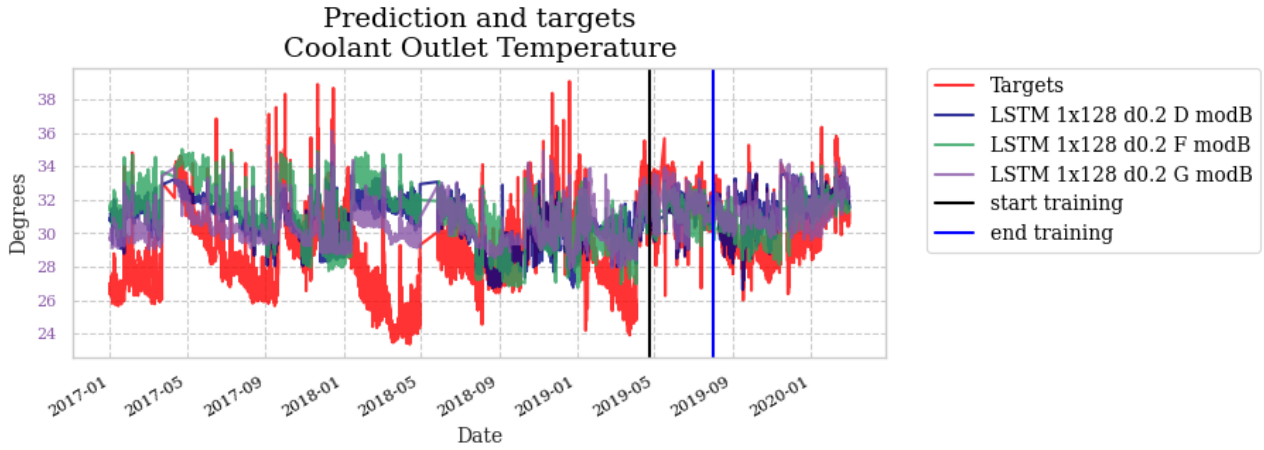


(c) Deviations for facility D using linear models trained on facilities D, F and G.

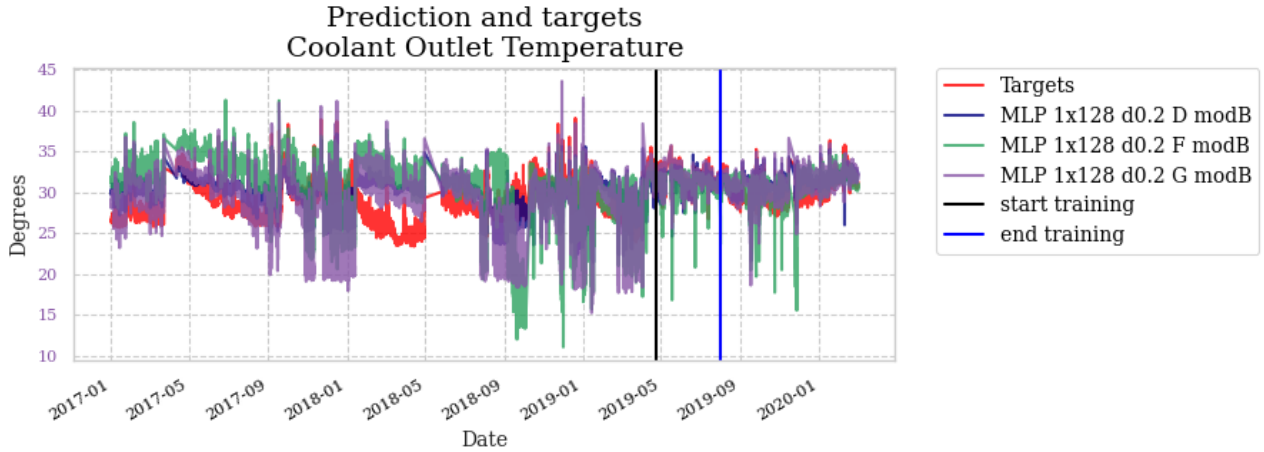
Figure 7.11

while also displaying large variations. The corresponding MLP model trained on dataset D shows a gradual decrease in predicted value with little variation. LSTM models show some of the same behavior, with gradually decreasing values, although with more variations than the linear model. The same can be said for linear models using predictive model B. Neither model correlates well with the model trained on dataset F.

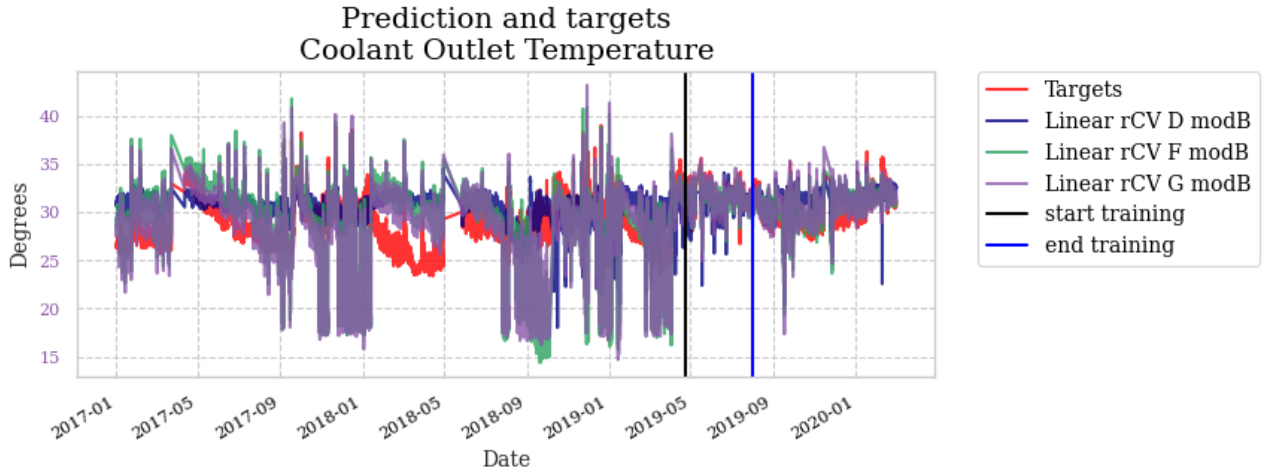
It is difficult to evaluate the model performances on facility G appropriately, because the coolant outlet temperature itself is decreasing in a pattern somewhat similar to the expected fouling pattern. This means almost any predictor, even one predicting a constant value, may result in a deviation graph which corresponds to the assumed fouling levels. Even when considering this, cross model prediction for facility G appears more promising than for the facilities previously discussed when using predictive model B. The calculated R^2 metrics are relatively high considering the models are trained on different datasets. Results are seen in Figures 7.12a, 7.12b and 7.12c. For MLP models, models trained on dataset D and F show less signs of overfitting to the valve opening value than the model trained on dataset G. All LSTM models show almost the same prediction pattern, although with some static differences. The deviation curves have very similar patterns. The predictions may become even more accurate if static differences are accounted for by adding a derived constant term to the predictions for each model. Linear models seem to have much of the same valve opening overfitting problems. Overall, results for dataset G show some promise.



(a) Predictions for facility G using LSTM models trained on facilities D, F and G.



(b) Predictions for facility G using MLP models trained on facilities D, F and G.



(c) Predictions for facility G using linear models trained on facilities D, F and G.

Figure 7.12

Chapter 8

Discussion

This chapter discusses the discoveries made in Chapter 6 and the results obtained in Chapter 7 in greater detail. Attempts are made to justify some of the decisions made throughout the thesis work, while highlighting possible consequences and areas of improvement. The discussion serves as a basis for recommended future work presented in the following chapter, in section 9.3.

8.1 Nature of heat exchanger data

As noted in section 6.3, outlier removal for heat exchanger data is not trivial. Especially for linear models, outliers affect the fitted model negatively. This is likely part of the reason why linear models proved to have poor performance for facility F, for which numerous outliers are present with nonzero mean. For trained models using loss metrics, a necessary condition to maintain model performance in the presence of outliers is that the reduction in loss when accurately fitting the real data exceeds the increase in loss due to poor fit for outlier values. For use with linear models, comprehensive outlier removal seems desirable. The same can be said for neural network models, although uniform time steps between each sample should be maintained as best as possible for recurrent models.

Interesting relations are found between $T_{c,i}$, $\dot{m}_c$ and predicted $T_{c,o}$ when using seawater as a coolant, such as for facility G. Questions were previous raised as to whether models could learn the effects of seasonal variation in $T_{c,i}$ using relatively short training periods. Neither model shows any signs of such variation in predicted $T_{c,o}$, which may indicate that models trained on a limited time period with little seasonal variation in $T_{c,i}$ are infeasible. In fact, linear models find a negative weight for the $T_{c,i}$ parameter for predictive model A, which appears illogical at first glance. However, when considering the coolant flow parameter, the relations found by the trained models are reasonable. The decrease in $T_{c,o}$ as a result of increased coolant flow rate requirements during summer balances out,

or even exceeds, the increase in seawater temperature. As such, it is considered that long training periods are not necessary to model the seasonal variation in $T_{c,i}$, because the short-term variations are sufficient in modelling the system relations. However, this does not mean the timing of the training period is indifferent. Seawater systems may behave differently during the summer in general, because the coolant demand is higher. Overall, periods with limited fouling, independent of seasonal behavior, are considered ideal for model training.

Values from real facilities are found to vary rapidly, resulting in rough graphs. This in itself does not necessarily mean that the model output parameter is a complex high-varying function of the input parameters, for which it was noted that deep architectures are preferable. The success of linear models for facility D and G seems to indicate that even though the function may be complex, it can be accurately estimated using simpler methods. Whether deep architectures are still preferable even if the function is just high-varying, and not necessary complex, is debatable. Even though the approximated function of neural networks may be more exact, the explainability is severely reduced compared to linear models.

Facilities are selected to cover some different operating conditions. However, this does not suffice to make conclusions for all possible heat exchanger systems. Facilities with different fluid compositions, upstream- or downstream equipment, and perhaps most importantly other types of fouling, may find different results applying the same methods as explored in this thesis. Even so, the derived schemes are considered quite generic, considering that the methodology is based on heat exchanger theory.

8.2 Applicability of unsupervised methods

Limited success was found using unsupervised methods, namely PCA and correlation analysis. Particularly the results obtained by observing the two-dimensional operating point of the systems using PCA proved difficult to interpret, because the principal components do not correspond to real sensor measurements in practice. Plotting and perceiving different periods of increasing fouling did not show any clear similarities. Although it seems possible to use PCA for tracking changes in operating point over time, it is difficult to relate these changes to increases in fouling. Overall, it is considered more beneficial to evaluate changes in operating point for two-dimensional plots consisting of existing dataset features, such as $\dot{m}_h$ and dP_h, or $\dot{m}_h$ and $T_{c,o}$.

No notable discoveries were made while doing correlation analysis, apart from a small difference in correlation between dP_c and $T_{c,i}$ for data with and without fouling. Without any similar facilities to compare these results with, nothing decisive can be said with regards to fouling. It can be concluded that, at least from a small selection of datasets, it is very difficult to determine fouling indicators simply by reading the correlation and correlation difference plots.

8.3 Applicability of supervised methods

Supervised methods in general are promising. Because labeling output data based on fouling factor or the presence of fouling is impossible, regression models must be used. Pressure difference was neglected in favor of temperature models because no comparative results were obtained using facility D. In general, training separate models with $T_{c,o}$, dP_h and dP_c as output parameters, respectively, may still be desirable in practice to account for a larger variety of fouling mechanisms.

Predictive model A is found sufficient in estimating fouling for simulated data using linear and MLP models, with slight increases in performance found by including additional parameters. The performance of LSTM models for dataset D is found to be very poor, although this is attributed to little training data without much variation, low sampling rate and small lookback parameter.

For facility F, linear and MLP models are found to have limited accuracy. This indicates that such models are not applicable for facilities under development with gradually increasing $\dot{m}_h$. Better performance, although still lackluster compared to LSTM models, is found using predictive model B. LSTM models are found to perform very well. For facility G, linear models with predictive models A and C are found to perform well, while the addition of other parameters cause overfitting. The performance of LSTM models is superior for all predictive models. Small improvements in calculated metrics are found through the use of ensemble models with LSTM and linear submodels, although not considered significant.

The lack of correlation between $v_{\%}$ and $\dot{m}_c$ when using centralized coolant systems, and the risk of drastic overfitting, indicate that $v_{\%}$ is not a viable model input feature for such systems. However, feature transformations like clipping may be applied to make $v_{\%}$ indicate $\dot{m}_c$ more accurately, although at the cost of introducing considerable uncertainty. For well behaved closed-loop coolant systems, as for facility F, the use of $v_{\%}$ may be more suitable. Such systems are generally found to be easier to model due to the lack of variation in coolant inlet temperature.

The use of predictive schemes in general seems to outperform the manual parameter inspection benchmark, because fouling is difficult to estimate using such inspections due to lack of appropriate reference values. Performance for benchmark regression models is found to be comparable to linear models. An exception is for tree-based models, which overfit significantly.

It can be argued that machine learning models are excess if predictive models C or E are applicable. In these models, all the necessary measurements to calculate heat exchanger effectiveness using traditional thermodynamics are present. However, as argued when formulating the thesis methodology, thermodynamic calculations introduce considerable uncertainty when estimating the fluid properties. If deviation between predicted and measured $T_{c,o}$ can be related to the fouling factor for arbitrary processing facilities, it is believed that regression models achieve more reliable results than traditional techniques.

Performing predictions with models trained on different datasets gives unsatisfying results in general. Even so, there are some chases for which the pattern of prediction are quite similar. This indicates that predictions using models trained on different datasets may perhaps be utilized in cases where the processing plants are known to have many similarities, which is not necessarily the case for facilities F and G analyzed here. Performing predictions in this manner is particularly relevant in cases where a dataset has no apparent training phase. In new facilities, this technique may be the only suitable manner of applying predictive models until adequate training data is acquired. Further investigation is required in order to determine the applicability of such predictions.

Note that Eq. 4.5, derived and used to justify the existence of the predictive models using $T_{c,o}$ as output parameters with a reduced number of input parameters, is based on equations for counterflow heat exchangers. More specifically, Eq. 3.22 used to derive Eq. 4.5 is only valid for counterflow arrangements. Equivalent formulas for parallel flow arrangements can be derived, although is omitted in this thesis because such heat exchangers are rarely used in practice for oil and gas processing. Normally, more advanced counterflow designs are used. For these, Eq. 4.5 may not be entirely accurate. Even so, equivalent formulas are likely to be obtainable also in these cases. Obtainability is the primary concern for machine learning algorithms, as the exact formulas are not needed.

8.4 Model evaluation

Model performance is estimated empirically because fouling factor is not measurable in practice. The decision not to use distinct testing data with similar properties as the training and validation data is considered unfortunate, although somewhat justified based on the argumentation presented in section 4.3.1. Appropriate use of testing data would have ensured an additional ranking criteria for trained models. However, with a limited number of data samples, validation and training splits would likely have had an influence on the obtained results. Using a traditional, random split between training, testing and validation data is recommended if larger datasets with higher sampling rates are obtained.

By using empirical analysis to evaluate performance, the achieved results are inherently speculative. Apparent model performance can be deceiving, leading to the selection of inappropriate models. Although this is largely avoided by thorough analysis of the model fit, additional metrics should ideally have been implemented to determine model applicability with a higher level of certainty. For instance, models can be manually evaluated using engineered testing samples, to decide whether the variation in output for certain input parameters is reasonable. By varying one input feature at the time, with the remaining features constant at reasonable values observed throughout the time series, the response in $T_{c,o}$ could have been observed. Techniques such as this may be used as additional evaluation metrics, although somewhat cumbersome and time-consuming.

8.5 Deep learning

Sampling rates of 10 and 30 minutes were used for the acquired data. For use in deep learning, the sampling rate ideally should have been higher, allowing larger datasets to be gathered for a short period of time, resulting in minimal fouling for the defined training periods. Additionally, the lookback parameter of recurrent neural networks may then be increased while maintaining the same lookback duration. These arguments highlight why high sampling rates and large amounts of data are desirable, even though the underlying fouling mechanisms mostly evolve slowly.

The lookback duration in itself is not easy to define. Because fouling accumulates over a long period of time under normal circumstances, the lookback duration cannot cover the entire fouling period. It is debatable how large lookback duration is optimal in practice. Long lookback duration helps smooth out outliers in the predicted values, enhancing the visibility of slowly evolving patterns. However, this also means outliers remain part of the prediction window for a longer period of time, affecting the predictive results. Maintaining the learning gradient becomes increasingly challenging for large lookback parameters, and training time for each model increases drastically. A decision is made to use 6 hours as lookback duration, although not backed by extensive trail and error. Use of more specific lookback durations could be tested, for instance 24 hours to account for the daily variation in temperatures, although unlikely to have an effect on the processing facilities in practice.

How much data must be acquired to fit or train appropriate models in practice is an open question. A benefit of long training periods is that numerous different operating conditions may be encountered throughout. However, use of such datasets is infeasible because the training period should have minimal levels of fouling. Based on the thesis results, short training periods are considered to contain sufficient variation to accurately model the system. Hence, high sampling rates are desirable to ensure large datasets can be gathered over a limited time span. There are no formal rules to determine how large datasets are required for the training of deep learning regression models. With a sampling rate of 1 minute, 1440 samples may be gathered each day. Considering the pros and cons, training periods of approximately a month appears reasonable, resulting just under 50.000 samples.

Although deep architectures are generally recommended in literature, such models proved difficult to train using small datasets and limited computing power. Further research using more powerful hardware, large datasets, and more advanced techniques like learning rate scheduling and weight initialization is advised. However, keep in mind that the model complexity increases while the explainability decreases when more advanced techniques are used. Even so, further reducing the explainability may not be as problematic considering that recurrent models are already difficult to explain.

8.6 Uncertainty

A particular strength of machine learning is that the derived models can be generic and adaptive, because limited or no domain knowledge must be applied when training models. However, it is somewhat ironic that the usability of such models in practice requires a high level of confidence that they accurately model the system domain. This can only be obtained through extensive empirical testing of the trained models, or by finding theoretical justification for the existence of such models. Generic and adaptive models may not be obtainable in practice.

Although techniques exist to evaluate the predictive uncertainty of trained models, this does not account for uncertainty in the modelling itself. This thesis argues that estimation of heat exchanger fouling in itself is so experimental that performing uncertainty analysis of the trained models is not a primary concern. This may not be the case for some other industrial applications where the degradation patterns are much easier to interpret and measure in practice. If uncertainty estimations for the trained models are to be derived, the method of estimating model uncertainty presented in section C.4 may be combined with uncertainty estimates for the data itself, like the measurement uncertainty.

8.7 Predictive maintenance

A relation between the added fouling factor and the calculated deviation using the predictive models is found for simulated data. However, this relation must be generalized in order to find accurate fouling estimates for arbitrary facilities. Equation 7.1, which was derived based on results for facility D, is most likely not applicable for other facilities due to differences in fluid composition, process and coolant flow rates, and other operating conditions. While a maximum deviation of -2 degrees is found for facility D, deviations as low as -6 and -7 are found for facilities F and G. Although the corresponding levels of fouling using the same relation as for facility D are not necessarily unrealistic, there exists no proof that these estimates are correct. Such proof may only be obtained if the fouling factor is measured during heat exchanger maintenance, which is likely not possible in practice. Alternatively, estimates can be derived using simulated data. A suggestion for how such estimates may be obtained is presented in section 9.3.2.

The thesis does not formulate any preventive maintenance scheme or derive cost based models. Hence, the achieved results may only be used for condition-based monitoring and maintenance, relying on fouling estimates obtained by using the predictive models. A master thesis or similar project focusing on predictive and preventive maintenance planning is a natural continuation of the work presented in this thesis.

Chapter 9

Conclusions

9.1 Thesis hypotheses

Based on review of heat exchanger and machine learning theory, two hypotheses regarding the monitoring of heat exchanger performance were formulated.

> **Hypothesis 1**: The sensors $\dot{m}_h$, $T_{h,i}$, $T_{h,o}$, $T_{c,i}$ and $T_{c,o}$ suffice to accurately estimate heat exchanger fouling using nonlinear regression models, based on deviation between predicted and measured $T_{c,o}$. Additional sensors $v_{\%}$, $\dot{m}_c$ and $P_{c,i}$ may increase the model accuracy at the expense of higher overfitting risk and increased model complexity.

Through the definition of a minimal predictive model A and subsequent fitting and evaluation of multiple regression models, the first part of hypothesis 1 concerning fouling estimation is considered **credible**. Results are presented for three different facilities, all for which considerable increases in thermal performance are identified following heat exchanger maintenance. Continuous performance degradation between each maintenance is also observed. An apparent linear relationship between added fouling factor and deviation in coolant outlet temperature is found for simulated data.

Furthermore, predictive models B through E were defined to investigate the effects of using additional sensors. Increased predictive performance is found by the inclusion of the $\dot{m}_c$ parameter. Meanwhile, clear overfitting is observed when $v_{\%}$ is included for facility G. $P_{c,i}$ is found to have limited variation, and thus system influence, in practice. As a result, the second part of hypothesis 1 concerning addition sensors is also considered **credible**.

Necessary assumptions are that coolant flow rate for the heat exchanger in question is controlled according to a desired process outlet temperature, and that operating conditions such as fluid compositions do not change substantially. The fitting of regression models requires data to be acquired with minimal levels of fouling.

> **Hypothesis 2**: In the presence of heat exchanger fouling, deviation between predicted and measured dP_h may indicate whether fouling is on the process or coolant side. If such deviation is found, at least some fouling is on the process side. Otherwise, fouling is on the coolant side.

Challenges were faced when defining the hydraulic effects of fouling for simulated data. As a result, no clear differences in pressure drop are observed for facility D with fouling on either side of the heat exchanger. Consequently, limited emphasis is placed on testing of this hypothesis. For facility G, no clear fouling patterns are discovered using predictive models with dP_h as output parameter. Still, hypothesis 2 is considered **plausible** based on heat exchanger theory. The measurement of pressure differences dP_h and dP_c may be important in practice to account for cases of fouling not reflected in heat exchanger thermal performance.

9.2 Additional remarks

Condition monitoring of heat exchangers using machine learning has proven to be a complicated subject, for which many considerations must be made. Most importantly, model performance must be evaluated empirically, because fouling factor is not measurable in practice. A relation between deviation in coolant outlet temperature and fouling factor is found for simulated heat exchanger data. This relation may not generalize well for arbitrary facilities with different heat exchanger designs, coolant systems, fluid compositions and flow rates. Efforts should be made to establish equivalent relations for a variety of operating conditions.

The sensors $\dot{m}_h$, $T_{h,i}$, $T_{h,o}$, $T_{c,i}$ and $T_{c,o}$ are found to be vital in evaluating heat exchanger performance. It is recommended that future facilities install these sensors, as a minimum. Increased performance for predictive models may be obtained through the addition of $\dot{m}_c$ in particular. Recurrent neural networks and linear regression models are found suitable for modelling heat exchanger data, and are recommended for further development. Excellent model performance is found for simulated data, with similar degradation patterns for real facilities.

Critically, the thesis methodology is based on theoretical definitions. Despite being referred to as a data-driven and hands-on discipline, the use of machine learning models in practice relies on the ability to convince supervising engineers that the trained models are in fact valid for the system domain. One way of doing so is by finding theoretical justification for the existence of such models, as is done here.

Although this thesis lays a solid foundation for the estimation of heat exchanger fouling using machine learning methods, and to facilitate further research within this topic, there is much work to be done before preventive maintenance models can be defined and used. Performing optimally timed preventive maintenance through the estimation of heat exchanger fouling requires the inclusion of cost and maintenance related models, which is omitted entirely in this thesis work. Recommendations for such work, and more, is presented in the following section.

9.3 Recommendations for further research

Recommendations for future work are covered quite extensively to facilitate and encourage further work within the same Equinor research project, and to tie up any loose ends in the wide range of subjects discussed throughout the thesis. During development of the thesis methodology, there has been numerous ideas for models, estimators and schemes that could not be tested due to various constraints, primary the already large extent of the thesis. These recommendations are formulated in this section, some in detail while others more vaguely.

9.3.1 Models and model training

Simulated training data

Simulated data was proven useful because fouling levels can be added explicitly to heat exchanger modules. However, the performed simulations were limited in size because manually defining realistic input parameters for large datasets is an extensive and difficult task. An interesting suggestion, although not tested in practice, is to use the values of real facilities as input to simulation models. For instance using predictive model A, values for the features $T_{c,i}$, $T_{h,i}$, $T_{h,o}$ and $\dot{m}_h$ from a real facility may be provided to a simulated model using a heat exchanger module with the same properties as the heat exchanger used in practice. Then, the resulting coolant outlet temperature $T_{c,o}$ without the presence of fouling can be simulated, from which a machine learning model can be trained. This ensures that training data can be acquired for which no fouling is present in the training data, making use of traditional validation and testing data feasible.

A drawback of the above approach is that the simulated output features are not measured values, but instead calculated by the defined simulation model. Hence, regression models fitted to this data will in practice only learn to produce the same transformation from input to output as the simulation program. Even so, using the trained machine learning model, e.g. for real-time facilities, is considered more convenient than running simulations in practice.

This method may also be applicable when deriving models for new facilities. Once the mean and variance of each feature is calculated or measured for a new facility, already acquired datasets may be transformed to match these operating conditions and used in simulations similarly as explained above. This allows for at least temporary predictive models to be fitted before sufficient data has been acquired for the new facility itself.

Linear basis function expansion

If found to be applicable, linear models are highly preferred as opposed to more complex models. The capabilities of such models using the proposed predictive models are considered promising in general, especially given that this thesis im-

plements very simple linear regression models. As several physical systems are known from theory to have nonlinear relations between some features, more accurate models are likely to be derived through the use of linear basis function expansion.

Basis function expansion may be performed by transforming existing features, or adding additional features which are the transformations of existing features. Imagine a dataset contains the parameters pressure drop dP, inlet temperature T_i, outlet temperature T_o, valve opening $v_{\%}$ and mass flow $\dot{m}$, for which input and output parameters of a predictive model may be chosen as desired. A quadratic term $\dot{m}^2$ would allow the fitting of much more accurate models for estimating pressure drop. Product terms such as $v_{\%} \cdot \dot{m}$ may prove useful. The change between subsequent measurements, e.g. $\dot{m}_t - \dot{m}_{t-1}$, may be added as a feature. Parameters prone to overfitting may be adjusted. For instance, the valve opening $v_{\%}$, which was found to have limited effect above approximately 60%, can be transformed, e.g. by flatting the value using a logarithmic transformation as follows:

$$v_{\%,new} = \begin{cases} v_{\%}, & \text{if } v_{\%} < 60 \\ 60 + k \log(v_{\%} - 60 + 1), & \text{otherwise} \end{cases} \tag{9.1}$$

Doing these types of feature additions or transformations is rather trivial using polynomial model features in a library like Scikit-Learn[1].

Deep learning models

LSTM models proved to be promising for the modelling of heat exchanger behavior. While numerous models and architectures are explored for the datasets acquired in this thesis, the use of deep learning on very large datasets remains untested. This proved infeasible due to data retrieval and hardware constraints. Although networks of one or two hidden layers with 64 or 128 units were found suitable in this thesis, deeper architectures may still prove useful for larger datasets. The sampling rate limitation of the Equinor servers is not known, although data with sampling rates of 1 minute, or perhaps even lower, should be possible to acquire.

When fitting models for deep architectures, more advanced schemes not explored in this thesis may be tested. This includes e.g. pre-training, learning rate scheduling, layer-wise training, and weight initialization. However, it should be noted that the added performance from using these techniques may be difficult to measure in practice, because the performance in general is difficult to measure. This problem may be solved by the use of large datasets, and thus possible definition of testing data with similar properties as the training and validation data. Although ensemble models were found to yield little improvement throughout this thesis work, further investigation of such models may also be worthwhile.

[1] `https://scikit-learn.org/stable/modules/linear_model.html#polynomial-regression-extending-linear-models-with-basis-functions`

Outlier removal

Outlier behavior for heat exchanger data was found to vary between facilities, e.g. showing completely different types of outliers for facilities F and G. A large quantity of outliers across all features was argued to occur due to drastic changes in process flow rate. More advanced techniques for replacing outliers and missing data may be applied, for instance using interpolation schemes or the Amelia algorithm. Another option, particularly for linear models, is to simply remove all apparent outlier rows through individual graph inspection. Because linear models require no sequences of data like recurrent neural networks do, apparent outliers may be removed rather trivially. Despite removing numerous outliers, large datasets and the variation in parameters during normal operation should ensure enough samples are kept to accurately fit linear models.

Uncertainty assessment

Appendix C.4 explains how dropout regularization may be used to find an approximated Gaussian distribution for the model uncertainty. However, no effort is put into explaining the underlying uncertainty of the acquired data and the irreducible noise. In order to assess the complete uncertainty of the applied methods, all sources of uncertainty must be included. Once a complete assessment of the method uncertainty is obtained, various statistical schemes may also be applied to the problem, as briefly discussed in section 3.2.6.

As argued in the thesis, although uncertainty estimates for the data and trained model are undeniably important, they may be irrelevant compared to the uncertainty introduced by the predictive and preventive maintenance modelling itself. Because the fouling phenomenon and its monitoring is not yet well defined, it is suggested that larger emphasis is put on finding suitable predictive schemes before assessing model uncertainty.

9.3.2 Predictive maintenance

Predictive maintenance modelling

This thesis takes great strides towards defining useful fouling estimators for simulated data, training predictive models and generalizing this for real facilities. However, no work is performed with regards to defining condition-based or predictive maintenance schemes. As was noted in section 3.2.6, finding the monitoring model is only part of the problem. What remains is implementing a suitable preventive maintenance scheme using this model.

A first step may be to determine the applicability of linearly relating the estimated $T_{c,o}$ deviation to the fouling level, and comparing the corresponding fouling level to design values. The relation between deviation in $T_{c,o}$ and fouling factor found for the simulated facility D may not generalize to different facility flow rates or

operating conditions. Initial research should seek to find maintenance data for which fouling factor is measured in practice, and compare this level with the estimated deviation. A similar scheme as suggested in section 9.3.1 may also prove useful, using data from real facilities as input parameters for a simulation model. For a heat exchanger module with similar properties as used in practice, different added fouling factors and corresponding changes in coolant outlet temperature may be investigated. Based on this, it may be possible to find reasonable estimates that relate $T_{c,o}$ deviation to fouling factor, even for real facility.

Using the acquired relations between fouling factor and changes in coolant outlet temperature, maintenance requirements may be defined as follows. If the required process outlet temperature cannot be reached, maintenance must be performed. Otherwise, if the estimated fouling factor exceeds the defined limitations, the need for maintenance must be evaluated. Additionally, predictive models using pressure difference may be used similarly.

Doing the above only derives a scheme applicable for condition-based monitoring based on predefined fouling thresholds. These thresholds may not be accurate in practice. For instance, the facility may operate acceptably despite reaching this level of fouling. To expand this into predictive maintenance, some model foreseeing the future accumulation of fouling and its effect on the facility must be defined. A suggestion for such a model is described below, in section 9.3.2.

Estimated limit until maximum coolant capacity

Because constant thresholds may be inadequate for maintenance planning in practice, a model which takes the limitations of the facility into account is suggested. When fouling accumulates, more coolant flow is required to reach the desired process outlet temperature. The supply of coolant is limited, hence the remaining amount of available coolant supply may be related to a corresponding maximum further increase in fouling. When this maximum level is reached, the desired process outlet temperature cannot be maintained, and so maintenance is required.

Coolant flow rate in itself is rarely measured, and the overall coolant supply may be difficult to estimate. Hence, the coolant valve opening, which is typically always measured, may be used to estimate how much of the available coolant supply is used at any point in time. Recall that for facility G, the valve opening was operated quite drastically without necessarily correlating with the coolant flow rate. Meanwhile for facility F, the valve opening operated within a limited area of operation. For the latter case, estimating a limit until maximum coolant capacity is reached may be applicable. Such an estimator should relate the estimated fouling level, for instance through the use of the predictive models suggested in this thesis, to the valve opening parameter. Note that adjustments must be made so that the valve opening value matches the corresponding coolant flow rate, e.g. using a transformation like the one defined in section 9.3.1. Other relevant factors are coolant inlet temperatures and seasonal dependencies.

Deriving cost models

Maintenance planning, and inherently the predictive maintenance scheme, is often related to the definition of cost models. Maintenance should be performed so that costs are minimized and system availability and performance is maximized. This can include performing maintenance for components which may not necessarily need it, because maintenance is performed for other connected components, stopping the production for that time period. A cost model should look to include the financial effects fouling has on the facility as a whole. If the increased coolant demand can be justified based on increased production rate, a certain level of fouling may be acceptable. However, if the production rate could be increased if fouling is reduced, performing maintenance may be financially viable. A practical starting point would be to investigate the financial burden of fouling problems, the cost and duration of heat exchanger maintenance, and the rate and extent of which maintenance is performed for connected components.

Identifying and handling sensor failures

The suggested predictive models do not account for sensor failures. If the sensor of an input or output feature fails, model predictions are likely to be altered greatly. Such failures would have to be identified by automatic sensor fault detection systems, or by supervising engineers through inspection of historical data. If sensor faults are difficult to detect using manual procedures, a fault may go unnoticed, and hence tamper with the predictive results of the applied monitoring models.

A real-time system to identify sensor failures is suggested, in which a series of regression models are fitted using different subsets of input and output parameters. The goal is a fault tolerant system that can identify whether the component in question or the measurement sensors are the cause of an indicated fault, by using differences in predicted values between the models. Exemplified by an arbitrary industrial component with input parameters i_1, i_2, i_3, and output parameter o_1 for some predictive model, the following regression models may be defined:

- **Model 1**: Predict o_1 using i_1, i_2, i_3
- **Model 2**: Predict i_1 using i_2, i_3, and potentially o_1
- **Model 3**: Predict i_2 using i_1, i_3, and potentially o_1
- **Model 4**: Predict i_3 using i_1, i_2, and potentially o_1

The corresponding deviations between predicted and measured value for each model should give useful information regarding the type of fault. In short, a deviation between predicted and measured value for model 1 combined with minor deviations for all remaining models would indicate a decrease in system performance. However, a sizable deviation for model 1 and exactly one other model predicting i_n, would indicate a sensor failure for that input sensor. Additional models with subsets of input and output parameters may be defined if the number of parameters is high.

Bibliography

[1] E. Jensen and T. Nordhus, 'Condition monitoring of heat exchangers in gas processing', Master's thesis, Norwegian University of Science and Technology, 2019.

[2] K. Hundman, V. Constantinou, C. Laporte, I. Colwell and T. Soderstrom, 'Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding', *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Jul. 2018. DOI: 10.1145/3219819.3219845. [Online]. Available: http://dx.doi.org/10.1145/3219819.3219845.

[3] I. Guyon and A. Elisseeff, 'An introduction to variable and feature selection', *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, Mar. 2003, ISSN: 1532-4435.

[4] T. G. Dietterich, 'Ensemble methods in machine learning', in *Proceedings of the First International Workshop on Multiple Classifier Systems*, ser. MCS '00, Berlin, Heidelberg: Springer-Verlag, 2000, pp. 1–15, ISBN: 3540677046.

[5] X. Qiu, L. Zhang, Y. Ren, P. Suganthan and G. Amaratunga, 'Ensemble deep learning for regression and time series forecasting', Dec. 2014. DOI: 10.1109/CIEL.2014.7015739.

[6] L. Deng, 'A tutorial survey of architectures, algorithms, and applications for deep learning', *APSIPA Transactions on Signal and Information Processing*, vol. 3, e2, 2014. DOI: 10.1017/atsip.2013.9.

[7] Y. Bengio, 'Learning deep architectures for ai', *Foundations*, vol. 2, pp. 1–55, Jan. 2009. DOI: 10.1561/2200000006.

[8] G. Hinton and R. Salakhutdinov, 'Reducing the dimensionality of data with neural networks', *Science (New York, N.Y.)*, vol. 313, pp. 504–7, Aug. 2006. DOI: 10.1126/science.1127647.

[9] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle and U. Montreal, 'Greedy layer-wise training of deep networks', vol. 19, Jan. 2007.

[10] Y. Bengio and Y. Lecun, 'Scaling learning algorithms towards ai', English (US), in *Large-scale kernel machines*, L. Bottou, O. Chapelle, D. DeCoste and J. Weston, Eds. MIT Press, 2007.

[11] A. Krizhevsky, I. Sutskever and G. Hinton, 'Imagenet classification with deep convolutional neural networks', *Neural Information Processing Systems*, vol. 25, Jan. 2012. DOI: 10.1145/3065386.

[12] C. Szegedy, A. Toshev and D. Erhan, 'Deep neural networks for object detection', Jan. 2013, pp. 1–9.

[13] O. Ronneberger, P. Fischer and T. Brox, 'U-net: Convolutional networks for biomedical image segmentation', in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells and A. F. Frangi, Eds., Cham: Springer International Publishing, 2015, pp. 234–241, ISBN: 978-3-319-24574-4.

[14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, 'Dropout: A simple way to prevent neural networks from overfitting', *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014, ISSN: 1532-4435.

[15] Y. Gal and Z. Ghahramani, *Dropout as a bayesian approximation: Representing model uncertainty in deep learning*, 2015.

[16] M. Ileby and E. Knutsen, 'Data-driven remote condition monitoring optimizes offshore maintenance, reduces costs', *World Oil*, pp. 60–62, 2017.

[17] J.-H. Shin and H.-B. Jun, 'On condition based maintenance policy', *Journal of Computational Design and Engineering*, vol. 2, no. 2, pp. 119–127, 2015, ISSN: 2288-4300. DOI: https://doi.org/10.1016/j.jcde.2014.12.006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2288430014000141.

[18] N. Görnitz, M. Kloft, K. Rieck and U. Brefeld, 'Toward supervised anomaly detection', *Journal of Artificial Intelligence Research (JAIR)*, vol. 45, Nov. 2012. DOI: 10.1613/jair.3623.

[19] S. Ahmad, A. Lavin, S. Purdy and Z. Agha, 'Unsupervised real-time anomaly detection for streaming data', *Neurocomputing*, vol. 262, pp. 134–147, 2017, Online Real-Time Learning Strategies for Data Streams, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2017.04.070. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231217309864.

[20] J. Gama, I. Žliobaitundefined, A. Bifet, M. Pechenizkiy and A. Bouchachia, 'A survey on concept drift adaptation', *ACM Comput. Surv.*, vol. 46, no. 4, Mar. 2014, ISSN: 0360-0300. DOI: 10.1145/2523813. [Online]. Available: https://doi.org/10.1145/2523813.

[21] V. Chandola, A. Banerjee and V. Kumar, 'Anomaly detection: A survey', *ACM Comput. Surv.*, vol. 41, no. 3, Jul. 2009, ISSN: 0360-0300. DOI: 10.1145/1541880.1541882. [Online]. Available: https://doi.org/10.1145/1541880.1541882.

[22] H. Zhao, H. Liu, W. Hu and X. Yan, 'Anomaly detection and fault analysis of wind turbine components based on deep learning network', *Renewable Energy*, vol. 127, pp. 825–834, 2018, ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2018.05.024. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0960148118305457.

[23] D. Hill and B. Minsker, 'Anomaly detection in streaming environmental sensor data: A data-driven modeling approach', *Environmental Modelling & Software*, vol. 25, pp. 1014–1022, Sep. 2010. DOI: 10.1016/j.envsoft.2009.08.010.

[24] O. Wu, A. Bouaswaig, S. Schneider, F. Leira, L. Imsland and M. Roth, 'Data-driven degradation model for batch processes: A case study on heat exchanger fouling', in. Jan. 2018, pp. 139–144, ISBN: 9780444642356. DOI: 10.1016/B978-0-444-64235-6.50026-7.

[25] Z. Tang and P. Fishwick, 'Feedforward neural nets as models for time series forecasting', *INFORMS Journal on Computing*, vol. 5, pp. 374–385, Nov. 1993. DOI: 10.1287/ijoc.5.4.374.

[26] H. Jiang, X. Li, H. Shao and K. Zhao, 'Intelligent fault diagnosis of rolling bearings using an improved deep recurrent neural network', *Measurement Science and Technology*, vol. 29, no. 6, p. 065 107, May 2018. DOI: 10.1088/1361-6501/aab945. [Online]. Available: https://doi.org/10.1088%2F1361-6501%2Faab945.

[27] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal and G. Shroff, 'Lstm-based encoder-decoder for multi-sensor anomaly detection', Jul. 2016.

[28] D. Park, S. Kim, Y. An and J.-Y. Jung, 'Lired: A light-weight real-time fault detection system for edge computing using lstm recurrent neural networks', *Sensors*, vol. 18, p. 2110, Jun. 2018. DOI: 10.3390/s18072110.

[29] S. Khan and T. Yairi, 'A review on the application of deep learning in system health management', *Mechanical Systems and Signal Processing*, vol. 107, pp. 241–265, 2018, ISSN: 0888-3270. DOI: https://doi.org/10.1016/j.ymssp.2017.11.024. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0888327017306064.

[30] B.-S. Yang, X. Di and T. Han, 'Random forests classifier for machine fault diagnosis', *Journal of Mechanical Science and Technology*, vol. 22, pp. 1716–1725, Sep. 2008. DOI: 10.1007/s12206-008-0603-6.

[31] A. Kumra, N. Rawal and P. Samui, 'Prediction of heat transfer rate of a wire-on-tube type heat exchanger: An artificial intelligence approach', *Procedia Engineering*, vol. 64, pp. 74–83, 2013, International Conference on Design and Manufacturing (IConDM2013), ISSN: 1877-7058. DOI: https://doi.org/10.1016/j.proeng.2013.09.078. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877705813015920.

[32] T. Ardsomang, J. Hines and B. Upadhyaya, 'Heat exchanger fouling and estimation of remaining useful life', *PHM 2013 - Proceedings of the Annual Conference of the Prognostics and Health Management Society 2013*, pp. 150–158, Jan. 2013.

[33] S. Lalot and H. Pálsson, 'Detection of fouling in a cross-flow heat exchanger using a neural network based technique', *International Journal of Thermal Sciences*, vol. 49, no. 4, pp. 675–679, 2010, ISSN: 1290-0729. DOI: https://doi.org/10.1016/j.ijthermalsci.2009.10.011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1290072909002373.

[34] M. Jerónimo, L. Melo, A. Braga, P. Ferreira and C. Martins, 'Monitoring the thermal efficiency of fouled heat exchangers: A simplified method', *Experimental Thermal and Fluid Science*, vol. 14, May 1997. DOI: 10.1016/S0894-1777(96)00146-X.

[35] F. Liporace and S. Oliveira, 'Real time fouling diagnosis and heat exchanger performance', *Heat Transfer Engineering*, vol. 28, Mar. 2007. DOI: 10.1080/01457630601064595.

[36] J. M. Campbell, *Gas Conditioning and Processing, Volume 1: The Basic Principles, Seventh edition*. Campbell Petroleum Series, 1992.

[37] J. M. Campbell, *Gas Conditioning and Processing, Volume 2: The Equipment Modules, Seventh edition*. Campbell Petroleum Series, 1992.

[38] T. Bergman, F. Incropera, D. DeWitt and A. Lavine, *Fundamentals of Heat and Mass Transfer*. Wiley, 2011, ISBN: 9780470501979. [Online]. Available: https://books.google.no/books?id=vvyIoXEywMoC.

[39] Maksim, *A typical phase diagram for a single-component material*, [Online; accessed 15-April-2020], 2006. [Online]. Available: https://commons.wikimedia.org/wiki/File:Phase-diag.svg.

[40] T. Bott, *Fouling of Heat Exchangers*. Elsevier Science B.V., 1995, ISBN: 0444821864.

[41] T. Bott and L. Melo, 'Fouling of heat exchangers', *Experimental Thermal and Fluid Science*, vol. 14, p. 315, May 1997. DOI: 10.1016/S0894-1777(96)00133-1.

[42] R. Shah and D. Sekulic, *Fundamentals of Heat Exchanger Design*. Wiley, 2003, ISBN: 9780471321712. [Online]. Available: https://books.google.no/books?id=beSXNAZblWQC.

[43] A. Barros, *Data Driven Prognostic and Predictive Maintance*, NTNU, Compendium of TPK4450 course, 2019.

[44] K. P. Murphy, *Machine learning: a probabilistic perspective*. Cambridge, MA: MIT Press, 2012.

[45] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[46] S. S. Haykin, *Neural Networks and Learning Machines*, ser. Neural networks and learning machines. Pearson Prentice Hall, 2009.

[47] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[48] F. Chollet, *Deep Learning with Python*, 1st. USA: Manning Publications Co., 2017, ISBN: 1617294438.

[49] T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, ser. Springer series in statistics. Springer, 2009.

[50] P. Brockwell and R. Davis, *Introduction to Time Series and Forecasting*, ser. Springer Texts in Statistics. Springer New York, 2006, ISBN: 9780387216577. [Online]. Available: https://books.google.no/books?id=P-bSBwAAQBAJ.

[51] D. K. Bhattacharyya and J. Kalita, *Network Anomaly Detection: A Machine Learning Perspective*. Apr. 2013.

[52] A. Ng, *Stanford cs229: Machine learning*, 2018. [Online]. Available: https://www.youtube.com/playlist?list=PLoROMvodv4rMiGQp3WXShtMGgzqpfVfbU.

[53] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan and D. Hassabis, 'A general reinforcement learning algorithm that masters chess, shogi, and go through self-play', *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018, ISSN: 0036-8075. DOI: 10.1126/science.aar6404. eprint: https://science.sciencemag.org/content/362/6419/1140.full.pdf. [Online]. Available: https://science.sciencemag.org/content/362/6419/1140.

[54] G. Hinton, 'To recognize shapes, first learn to generate images', *Progress in brain research*, vol. 165, pp. 535–47, Feb. 2007. DOI: 10.1016/S0079-6123(06)65034-6.

[55] E. Keogh and A. Mueen, 'Curse of dimensionality', in *Encyclopedia of Machine Learning and Data Mining*, C. Sammut and G. I. Webb, Eds. Boston, MA: Springer US, 2017, pp. 314–315, ISBN: 978-1-4899-7687-1. DOI: 10.1007/978-1-4899-7687-1_192. [Online]. Available: https://doi.org/10.1007/978-1-4899-7687-1_192.

[56] G. Cybenko, 'Approximation by superpositions of a sigmoidal function', *Mathematics of Control, Signals, and Systems (MCSS)*, vol. 2, no. 4, pp. 303–314, Dec. 1989, ISSN: 0932-4194. DOI: 10.1007/BF02551274. [Online]. Available: http://dx.doi.org/10.1007/BF02551274.

[57] J. Barata and M. Hussein, 'The moore-penrose pseudoinverse. a tutorial review of the theory', *Brazilian Journal of Physics*, vol. 42, Oct. 2011. DOI: 10.1007/s13538-011-0052-z.

[58] J. Duchi, E. Hazan and Y. Singer, 'Adaptive subgradient methods for online learning and stochastic optimization', *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, Jul. 2011.

[59] T. Tieleman and G. Hinton, *Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude*, COURSERA: Neural Networks for Machine Learning, 2012.

[60] D. Kingma and J. Ba, 'Adam: A method for stochastic optimization', *International Conference on Learning Representations*, Dec. 2014.

[61] Ixnay, *A diagram for a one-unit long short-term memory (lstm)*, [Online; accessed 20-April-2020], 2017. [Online]. Available: https://commons.wikimedia.org/wiki/File:Long_Short-Term_Memory.svg.

[62] Ixnay, *A diagram for a one-unit gated recurrent unit (gru)*, [Online; accessed 20-April-2020], 2017. [Online]. Available: https://en.wikipedia.org/wiki/File:Gated_Recurrent_Unit.svg.

[63] Michela Massi, *Autoencoder schema*, [Online; accessed 20-April-2020], 2019. [Online]. Available: https://commons.wikimedia.org/wiki/File:Autoencoder_schema.png.

[64] G. LLC. (2010). Kaggle, [Online]. Available: https://www.kaggle.com/ (visited on 30/04/2020).

[65] D. H. Wolpert, 'The lack of a priori distinctions between learning algorithms', *Neural Computation*, vol. 8, no. 7, pp. 1341–1390, 1996. DOI: 10.1162/neco.1996.8.7.1341. eprint: https://doi.org/10.1162/neco.1996.8.7.1341. [Online]. Available: https://doi.org/10.1162/neco.1996.8.7.1341.

[66] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu and X. Zheng, 'Tensorflow: A system for large-scale machine learning', in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283. [Online]. Available: https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf.

[67] H. W. Horn, *Condition Monitoring of Heat Exchangers using Machine Learning*, version 1.0, 2020. [Online]. Available: https://github.com/hermanwh/master-thesis.

[68] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009, ISBN: 1441412697.

[69] F. Chollet *et al.* (2015). Keras, [Online]. Available: https://github.com/fchollet/keras.

[70] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, 'Scikit-learn: Machine learning in python', *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[71] J. D. Hunter, 'Matplotlib: A 2d graphics environment', *Computing in science & engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[72] W. McKinney *et al.*, 'Data structures for statistical computing in python', in *Proceedings of the 9th Python in Science Conference*, Austin, TX, vol. 445, 2010, pp. 51–56.

[73] M. Waskom, O. Botvinnik, D. O'Kane, P. Hobson, S. Lukauskas, D. C. Gemperline, T. Augspurger, Y. Halchenko, J. B. Cole, J. Warmenhoven, J. de Ruiter, C. Pye, S. Hoyer, J. Vanderplas, S. Villalba, G. Kunter, E. Quintero, P. Bachant, M. Martin, K. Meyer, A. Miles, Y. Ram, T. Yarkoni, M. L. Williams, C. Evans, C. Fitzgerald, Brian, C. Fonnesbeck, A. Lee and A. Qalieh, *Mwaskom/seaborn: V0.8.1 (september 2017)*, version v0.8.1, Sep. 2017. DOI: 10.5281/zenodo.883859. [Online]. Available: https://doi.org/10.5281/zenodo.883859.

[74] T. E. Oliphant, *A guide to NumPy*. Trelgol Publishing USA, 2006, vol. 1.

[75] S. Chacon and B. Straub, *Pro git*. Apress, 2014.

[76] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla and C. Willing, 'Jupyter notebooks – a publishing format for reproducible computational workflows', in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, F. Loizides and B. Schmidt, Eds., IOS Press, 2016, pp. 87–90.

[77] H. W. Horn, *Condition monitoring of heat exchangers using machine learning*, NTNU, project thesis for the TMM4560 course, 2019.

[78] J. Honaker, G. King and M. Blackwell, 'Amelia ii: A program for missing data', *Journal of Statistical Software*, vol. 45, no. 7, pp. 1–47, 2011.

Appendix A

Additional theory

A.1 Peng-Robinson EoS

The Peng-Robinson equation of state is seen in Eq. 3.4. The parameters a, b and α are defined as follows. ω is the acentric factor. T_c and p_c are the critical temperature and pressure, meaning the end point of a phase equilibrium curve for the fluid. T_r is the reduced temperature.

$$a \approx 0.45724 \frac{R^2 T_c^2}{p_c} \tag{A.1}$$

$$b \approx 0.07780 \frac{R T_c}{p_c} \tag{A.2}$$

$$\alpha = (1 + \kappa(1 - T_r^{0.5}))^2 \tag{A.3}$$

$$\kappa \approx 0.37464 + 1.542265\omega - 0.26992\omega^2 \tag{A.4}$$

$$T_r = \frac{T}{T_c} \tag{A.5}$$

Appendix B

Preprocessing code examples

B.1 Data extraction using Python module

Code listing B.1: Data retrieval example

```
import pandas as pd
import dbmodule
from configs import (getConfig,getFileDirr)

def getData(component, start, end, resolution):
    (field,
    client,
    tags,
    tagsShort,
    tagDescriptions,
    tagUnits,
    timeframe,
    alias) = getConfig(component)

    if field is None:
        print("Invalid componenet " + component)
    else:
        con = dbmodule.Client(field, client)
        con.cache = None
        con.connect()

        df = con.read_tags(tags, start, end, resolution)

        dirr = getFileDirr(alias, start, end)
        df.to_csv(dirr, index=True)
```

B.2 Preprocessing using manual feature limitations

Code listing B.2: Data preprocessing example 1

```
import pandas as pd
from configs import (getConfig)
from utilities import (readDataFile, getDataWithTimeIndex)

# Preprocessing by feature limitations
# For each feature, an upper and/or lower
#     limit is defined to remove outliers
def preprocess(filename):
    component = filename.split('/')[-2]
    (field,
    client,
    tags,
    tagsShort,
    tagDescriptions,
    tagUnits,
    timeframe,
    alias) = getConfig(component)

    df = readDataFile(filename)
    df = getDataWithTimeIndex(df)
    df = df.dropna()

    df = df.loc[df['PDI0064'] > 0.5]
    df = df.loc[df['TI0066'] > 20]
    df = df.loc[df['TI0066'] < 40]
    df = df.loc[df['TZI0012'] > 100]
    df = df.loc[df['FI0010'] > 3000]
    df = df.loc[df['TT0025'] > 5]
    df = df.loc[df['TT0025'] < 14]
    df = df.loc[df['TT0026'] > 20]
    df = df.loc[df['PI0001'] > 6]
    df = df.loc[df['FI0027'] > 60]
    df = df.loc[df['TIC0022U'] > 1]
    df = df.loc[df['PDT0024'] > 1]

    resampleStr = "10min"
    df = df.resample(resampleStr).mean()
    df = df.dropna()
    df = df.rename(columns=tagsShort)

    filesplit = filename.split('/')
    filesplit[-1] = filesplit[-1].split('.')[0] + '_' + resampleStr + '.csv'
    dirr = '/'.join(filesplit)
    df.to_csv(dirr, index=True)
```

B.3 Preprocessing using quantile limitations

Code listing B.3: Data preprocessing example 2

```
import pandas as pd
from configs import (getConfig)
from utilities import (readDataFile, getDataWithTimeIndex)

# Preprocessing by quantile limitations
# For each feature, an upper and/or lower
#     quantile is defined to remove outliers
def preprocess(filename, low=0.03, high=0.97):
    component = filename.split('/')[-2]
    (field,
    client,
    tags,
    tagsShort,
    tagDescriptions,
    tagUnits,
    timeframe,
    alias) = getConfig(component)

    df = readDataFile(filename)
    df = getDataWithTimeIndex(df)
    df = df.dropna()
    df = df.loc[~df.index.duplicated(keep='first')]

    quant_df = df.quantile([low, high])
    df = df.apply(
        lambda x:x[
            (
                x >= quant_df.loc[low,x.name]
            ) &
            (
                x <= quant_df.loc[high,x.name]
            )
        ], axis=0
    )

    df = df.dropna()
    df = df.rename(columns=tagsShort)

    filesplit = filename.split('/')
    filesplit[-1] = filesplit[-1].split('.')[0] + '_processed.csv'
    dirr = '/'.join(filesplit)
    df.to_csv(dirr, index=True)

    print('File␣saved␣at␣' + dirr)
    print("␣")
```

Appendix C

Machine learning architecture comparisons

In this appendix, comparisons between different types of regularization and architectures for multilayer perceptron and recurrent neural networks are presented. This expands on the presentation of selected hyperparameters provided in section 4.3.4. The comparisons are included to demonstrate the effect of changing the model parameters and architectures, as well as to justify the final choice of hyperparameters. A series of Jupyter Notebooks accompany these results. A link is provided for each notebook in the footnote of the corresponding section. It proved infeasible to include all relevant graphs and results in the thesis itself, e.g. convergence graphs. Hence, viewing the notebooks in relation to each section is advised.

C.1 Loss metric

Calculated metrics for facilities D, F and G are seen in Table C.1, C.2 and C.3, respectively. Be reminded that the MAE and MSE loss values cannot be compared directly. Additional results and plots can be seen on GitHub[1].

Because MLP and LSTM model were found to have poor performance for facility D, less emphasis is placed on these scores. However, note that MSE generally outperforms MAE for this dataset, particularly when using predictive model B. A likely cause is that dataset D contains no outliers, for which squared error metrics typically perform well. For facility F, a substantial increase in performance is found wen using MAE for predictive model A with MLP models. Recall that dataset F has considerable outliers for process flow rate measurements. The difference is very small for the remaining models, indicating that using more advanced architectures

[1] https://github.com/hermanwh/master-thesis/blob/master/3_mae_mse_comp.ipynb

or the addition of more features may nullify the difference between absolute and squared metrics. For facility G, the performance is once more quite similar,

Similar behaviour is found for the trained models using each of the loss metrics, with a notable exception for the MLP models of facility F. Figures C.1a and C.1b show the predictions and deviations for the trained models using predictive model A. As seen in Figure C.1c, the MLP model with MSE fails to improve from its initial validation loss. The appearance of the convergence graph, with drastically lower validation loss in the first epoch, makes it likely that the first validation split has achieved an artificially low loss score. Even so, the following iterations fail to find validation loss lower than that of the MAE model. For low loss values, the mean squared metric should be considerably lower than the absolute metric, as is seen for the remaining facilities and models. Therefore, the MLP model using MSE for facility F is unlikely to outperform the model using MAE, even with a more appropriate initial validation split.

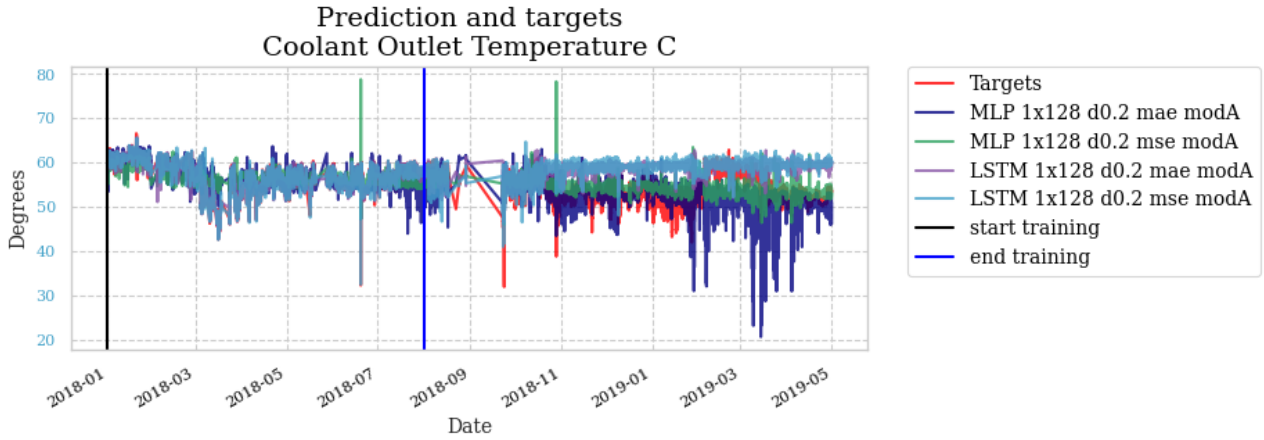
It is concluded that absolute metrics are more suitable than squared metrics. The use of MAE is likely to cause less issues with regards to outliers, while providing acceptable performance. Using several loss metrics during model evaluation is not advised, as comparisons between trained models becomes challenging.

Table C.1: Loss metric comparison for dataset D

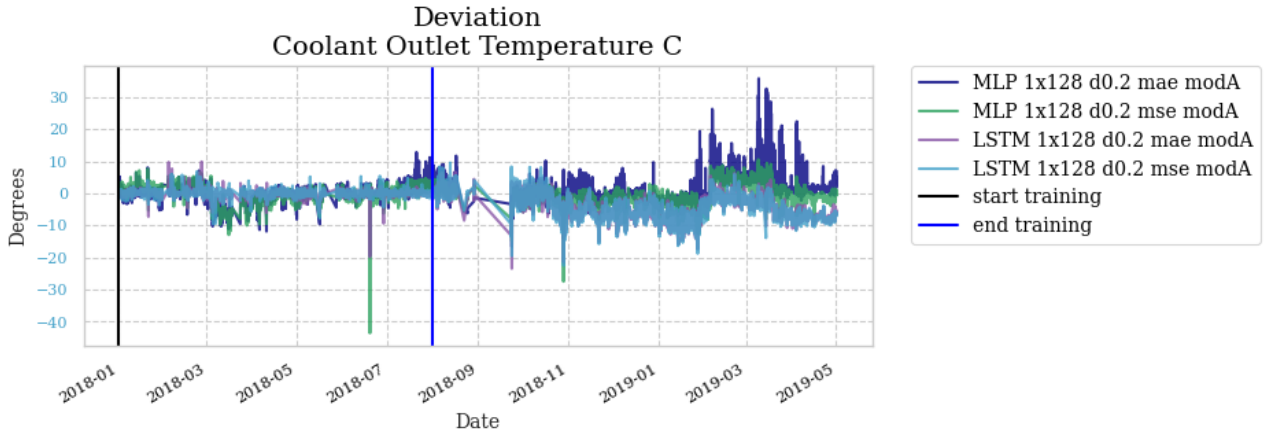
Model	Type	Metric	Min.loss	Loss	Val.loss	R^2
A	MLP	MAE	0.0946	0.1161	0.0445	0.9973
A	MLP	MSE	0.1527	0.0186	0.0033	0.9970
A	LSTM	MAE	0.0868	0.1277	0.1156	0.9444
A	LSTM	MSE	0.0055	0.0071	0.0443	0.9854
B	MLP	MAE	0.1288	0.2806	0.0443	0.9029
B	MLP	MSE	0.0182	0.0257	0.0019	0.9980
B	LSTM	MAE	0.0964	0.1289	0.1201	0.9350
B	LSTM	MSE	0.0072	0.0094	0.0381	0.9847

Table C.2: Loss metric comparison for dataset F

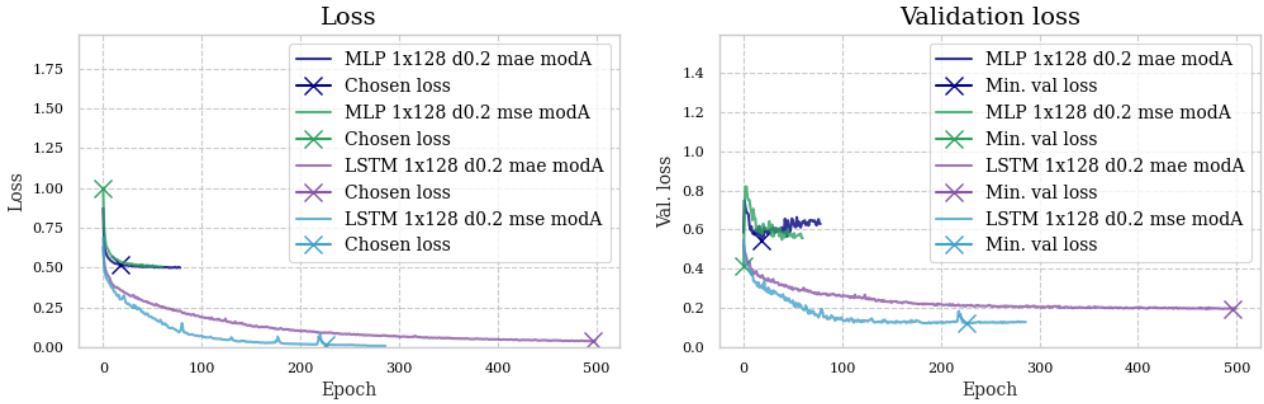
Model	Type	Metric	Min.loss	Loss	Val.loss	R^2
A	MLP	MAE	0.4978	0.5192	0.5461	0.4430
A	MLP	MSE	0.5035	0.9958	0.4137	0.2869
A	LSTM	MAE	0.0369	0.0374	0.1938	0.9683
A	LSTM	MSE	0.0072	0.0155	0.1208	0.9654
B	MLP	MAE	0.3197	0.4066	0.4884	0.7019
B	MLP	MSE	0.1858	0.2813	0.4469	0.7116
B	LSTM	MAE	0.0362	0.0436	0.1467	0.9743
B	LSTM	MSE	0.0072	0.0169	0.0711	0.9740



(a) Predicted and measured values.



(b) Deviation between predicted and measured values.



(c) Model convergence.

Figure C.1: Loss metric comparisons for dataset F with predictive model A.

Table C.3: Loss metric comparison for dataset G

Dataset	Type	Metric	Min.loss	Loss	Val.loss	R^2
A	MLP	MAE	0.4155	0.4780	0.4961	0.6060
A	MLP	MSE	0.3084	0.4057	0.4112	0.6120
A	LSTM	MAE	0.1621	0.2388	0.3126	0.8452
A	LSTM	MSE	0.0610	0.1374	0.2291	0.8504
B	MLP	MAE	0.1608	0.1644	0.1726	0.9523
B	MLP	MSE	0.056 49	0.0923	0.1123	0.9287
B	LSTM	MAE	0.1471	0.2180	0.3020	0.8559
B	LSTM	MSE	0.0423	0.1013	0.1865	0.8867

C.2 Multilayer perceptron

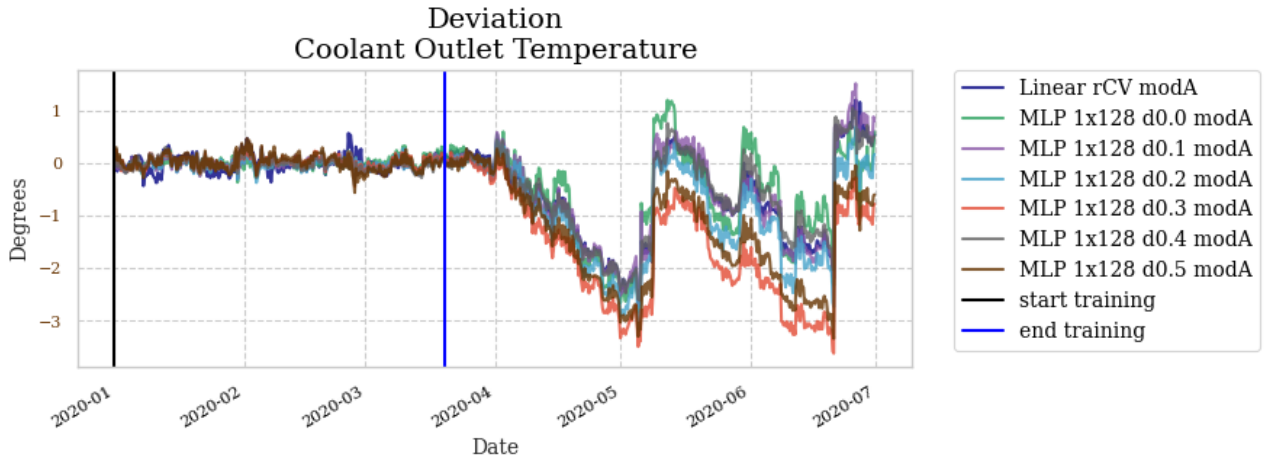
C.2.1 MLP dropout regularization

Calculated R^2 metric for facilities D, F and G are seen in Table C.4. Additional results and plots can be seen on GitHub[2]. Comparable performance is found between different dropout rates. By inspecting the prediction and deviation graphs, it becomes apparent that some rates have converged to worse solutions than others. Figures C.2a and C.2b show the calculated deviations for facility D using predictive model A and B, respectively. Models with dropout rates of 0.0, 0.3 and 0.4 show some clear shortcomings, either failing to accurately predict the training data, or making illogical predictions compared to the expected fouling pattern. Figure C.2c shows that the model using a dropout rate of 0.5 has converged to a very different solution than e.g. the model using 0.1 or 0.2. Overall, it is found that a dropout rate of 0.2 ensures satisfying regularization effects while converging quickly.

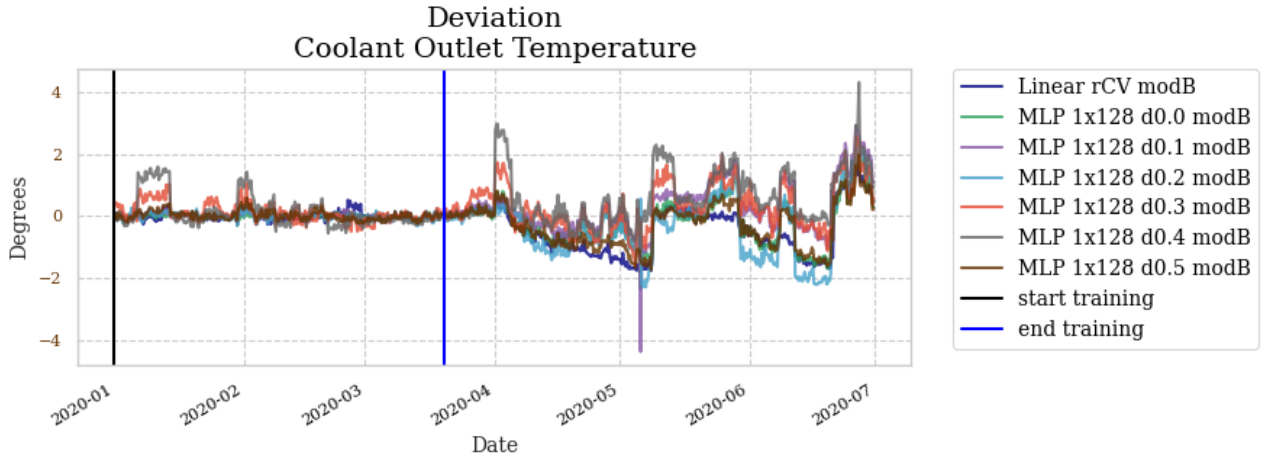
Table C.4: MLP dropout comparison for dataset D, F and G

	R^2 score model A			R^2 score model B		
Dropout rate	D	F	G	D	F	G
0.0	0.9947	0.4482	0.6065	0.9991	0.7345	0.9562
0.1	0.9979	0.2860	0.6072	0.9981	0.7181	0.9541
0.2	0.9974	0.4342	0.6775	0.9972	0.7187	0.9454
0.3	0.9973	0.4196	0.5964	0.9794	0.6934	0.9512
0.4	0.9968	0.2512	0.6186	0.9432	0.6999	0.9503
0.5	0.9955	0.4200	0.6545	0.9959	0.7062	0.9474

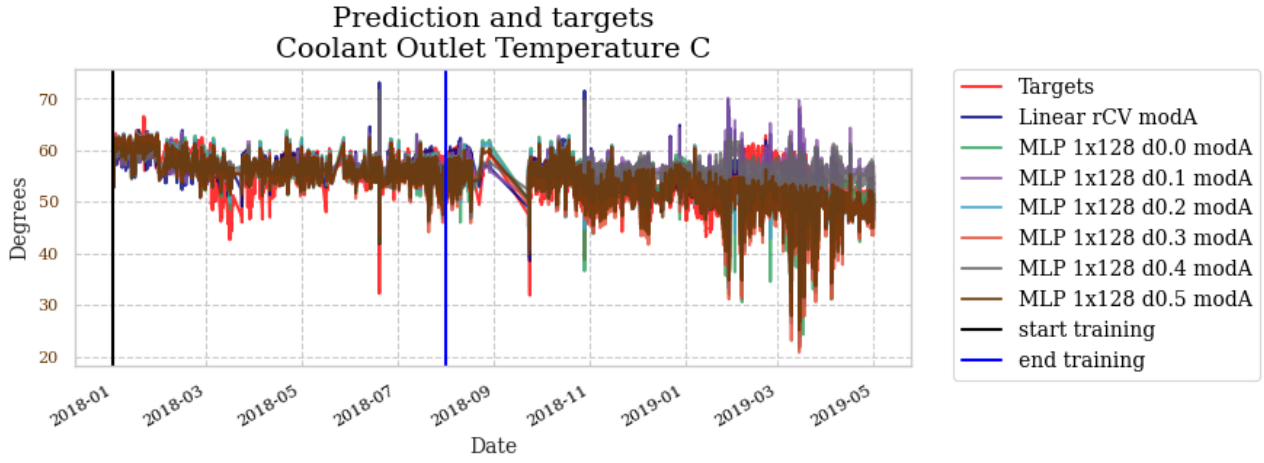
[2] https://github.com/hermanwh/master-thesis/blob/master/3_mlp_dout_comp.ipynb



(a) Deviation between predicted and measured values for dataset D model A.



(b) Deviation between predicted and measured values for dataset D model B.



(c) Predicted and measured values for dataset F, model A.

Figure C.2: MLP dropout comparison.

C.2.2 MLP LASSO regularization

Calculated metrics for facilities D, F and G are seen in Table C.5, C.6 and C.7, respectively. Additional results and plots can be seen on GitHub[3].

As seen in Figure C.3a, models with high amounts of regularization may be incapable of learning from the training data. This happens when the weight penalty term outweighs the error term whenever any weights are increased. Hence, such models converge to or close to the dataset mean, as observed for the models with a LASSO regularization rate of 1.0 and 0.5.

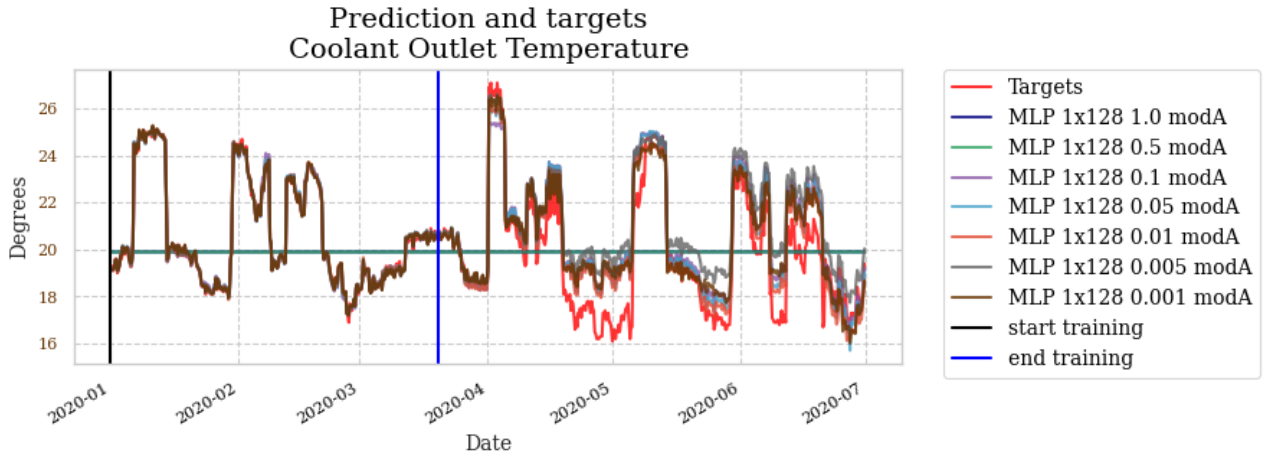
Figure C.3b shows the deviation between predicted and measured value for models using a regularization rate of 0.005 and 0.001 for dataset D. While both models have converged with similar loss and R^2 metrics, the deviation during the testing phase is quite different. The differences are particularly noticeable at the end of the time series after the second reset in fouling, with a deviation between the two models of more than 1 degree. This indicates that although models achieve similar metrics, their performance on the test data may be very different. A similar relation can be seen in Figure C.3c, although not as prominent. Some deviation between the predicted values of different models can be seen, particularly between June 2017 and May 2018.

A small amount of LASSO regularization, e.g. with a rate of 0.005 or 0.001, is found to achieve the best performance. The chosen validation loss for each rate is rather close for all models. However, the model convergence is at times much slower than for dropout regularization. Dropout regularization also achieves a final validation loss closer to the achieved training loss. This indicates less overfitting than for the models using LASSO regularization. As performance is quite similar in general, dropout is chosen as the preferred regularization technique.

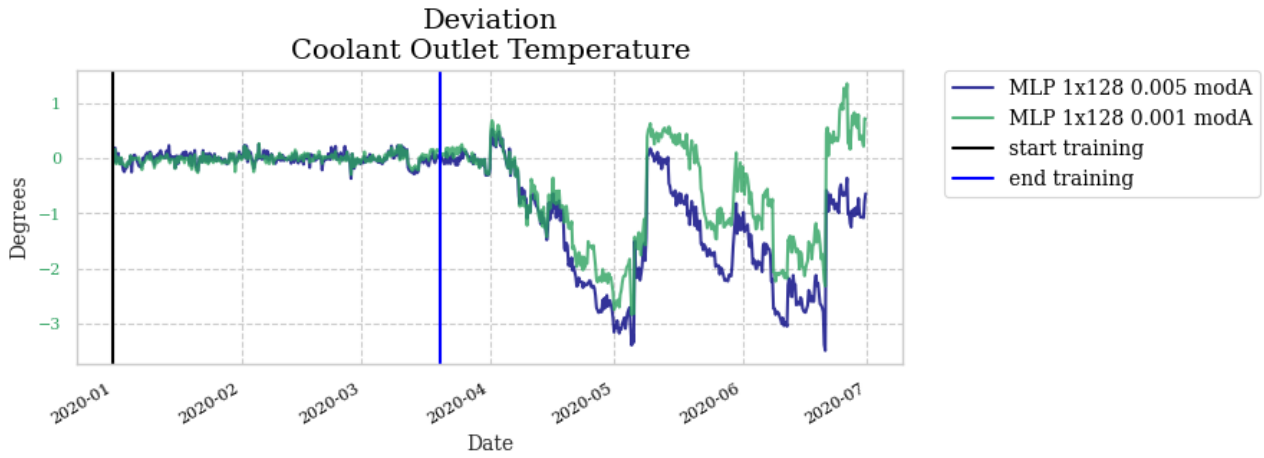
Table C.5: MLP regularization comparison metrics for dataset D

	Model A			Model B		
Rate	Loss	Val.loss	R^2	Loss	Val.loss	R^2
L1 1.0	0.9369	0.3812	0.1196	0.9465	0.3883	0.1195
L1 0.5	0.9228	0.3663	0.1196	0.9244	0.3694	0.1153
L1 0.1	0.1021	0.0987	0.9968	0.0901	0.0856	0.9979
L1 0.05	0.0738	0.0766	0.9976	0.0767	0.0784	0.9979
L1 0.01	0.0511	0.0613	0.9982	0.0451	0.0490	0.9991
L1 0.005	0.0588	0.0732	0.9980	0.0413	0.0462	0.9991
L1 0.001	0.0502	0.0688	0.9984	0.0402	0.0494	0.9991

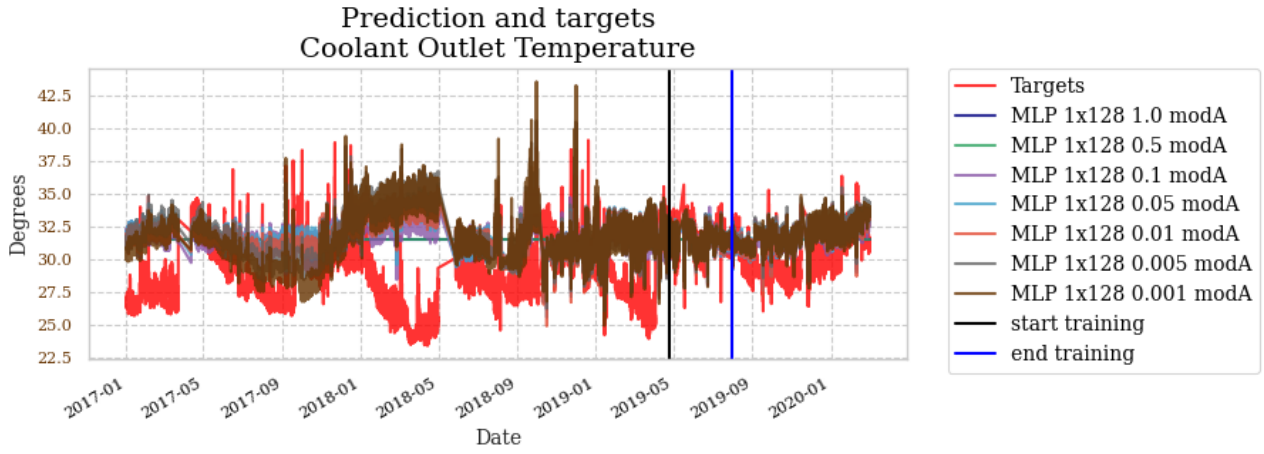
[3] https://github.com/hermanwh/master-thesis/blob/master/3_mlp_regu_comp.ipynb



(a) Predicted values for models with predictive model A.



(b) Deviation between predicted and measured value for models with predictive model A.



(c) Predicted values for models with predictive model A.

Figure C.3: MLP regularization comparison.

Table C.6: MLP regularization comparison metrics for dataset F

	Model A			Model B		
Rate	Loss	Val.loss	R^2	Loss	Val.loss	R^2
L1 1.0	0.9060	0.4646	0.0008	0.9147	0.4683	0.0002
L1 0.5	0.8901	0.4476	0.0006	0.8945	0.4470	0.0013
L1 0.1	0.8676	0.6225	0.1609	0.5153	0.5631	0.6067
L1 0.05	0.5856	0.5709	0.4152	0.4366	0.4936	0.6912
L1 0.01	0.5419	0.5467	0.4175	0.3857	0.4813	0.7550
L1 0.005	0.5282	0.5337	0.4256	0.3517	0.4807	0.7765
L1 0.001	0.4937	0.5481	0.4822	0.3890	0.5090	0.7360

Table C.7: MLP regularization comparison metrics for dataset G

	Model A			Model B		
Rate	Loss	Val.loss	R^2	Loss	Val.loss	R^2
L1 1.0	0.9005	0.6186	0.0125	0.9080	0.6294	0.0129
L1 0.5	0.8844	0.6059	0.0129	0.8889	0.6117	0.0118
L1 0.1	0.5375	0.5296	0.5811	0.3465	0.4418	0.9134
L1 0.05	0.5149	0.5936	0.6040	0.3098	0.4339	0.9136
L1 0.01	0.5099	0.6108	0.6027	0.1914	0.2908	0.9348
L1 0.005	0.4550	0.6018	0.6384	0.1759	0.2473	0.9392
L1 0.001	0.4328	0.5525	0.6791	0.1559	0.1748	0.9537

C.2.3 MLP architecture

The architecture comparisons presented in this section intend to demonstrate that different architectures may converge to very different solutions, despite all other hyperparameters and training data remaining the same. Therefore, determining a general approach that works for a large number of use cases is very challenging, or even impossible. An architecture must be chosen which has a sufficient number of layers and neurons not to underfit, without excess parameters to avoid overfitting, while keeping training time complexity manageable. The applicable hidden layer sizes may be limited by the available hardware. Large networks generally require large amounts of data to avoid overfitting. Network architecture optimization in general is not yet well understood, hence it remains an empirical procedure.

A limited grid search of one and two hidden layers is performed. Hyperparameters known to work well on the relevant datasets are chosen. Predictive model A is used. Calculated metrics for facilities D, F and G are seen in Table C.8. Additional results and plots can be seen on GitHub[4].

[4] `https://github.com/hermanwh/master-thesis/blob/master/3_mlp_arch_comp.ipynb`

Models with 16 neurons in each layer are generally found to have too little capacity, with higher validation loss and lower R^2 score than the more complex models of 64 and 128 neurons. Models with two hidden layer achieve higher performance for dataset F than models with a single hidden layers, while the opposite is found for dataset G. It is difficult to know in general whether the increase in model performance for higher capacity model is due to better model fit or model overfitting. Recall that models with many hidden layers and a large number of units in each layer have a large number of parameters, making such models capable of remembering the data rather than learning to generalize.

Training of models with multiple hidden layers is found to be more time consuming than single layer architectures. For dataset F in particular, the models with two hidden layers require between two and three times as many epochs to reach its minimal validation loss, while also requiring more time for each individual epoch. Training times for facility G are more similar between the different architectures.

Table C.8: MLP architecture comparison metrics, datasets D, F and G

Dataset	Layers	Min.loss	Loss	Val.loss	R^2
D	1x 16	0.4050	0.6087	0.1030	0.5937
D	1x 32	0.1905	0.2177	0.0498	0.9884
D	2x 16	0.3215	0.4058	0.0845	0.9276
D	2x 32	0.2072	0.2497	0.0535	0.9853
D	1x 64	0.1194	0.1473	0.0453	0.9955
D	1x128	0.0758	0.0865	0.0388	0.9978
D	2x 64	0.1192	0.1476	0.0457	0.9966
D	2x128	0.1161	0.1495	0.0394	0.9962
F	1x 16	0.3968	0.8282	0.5507	0.1722
F	1x 32	0.3598	0.4398	0.5657	0.6712
F	2x 16	0.3605	0.4407	0.4889	0.7364
F	2x 32	0.2702	0.2807	0.4529	0.8345
F	1x 64	0.3364	0.4172	0.4854	0.7025
F	1x128	0.3185	0.3906	0.4956	0.7124
F	2x 64	0.2484	0.2602	0.4589	0.8424
F	2x128	0.2160	0.2237	0.4369	0.8621
G	1x 16	0.4627	0.4721	0.5262	0.6375
G	1x 32	0.4499	0.5556	0.4839	0.5594
G	2x 16	0.4616	0.5114	0.5081	0.6118
G	2x 32	0.3975	0.4596	0.5717	0.6521
G	1x 64	0.4348	0.4942	0.5028	0.6095
G	1x128	0.3968	0.4109	0.5224	0.6795
G	2x 64	0.3873	0.5095	0.4935	0.6096
G	2x128	0.3587	0.5153	0.4986	0.5990

As seen in figures C.5a and C.5b, the deviation pattern for dataset D is quite similar for all models, apart from the models using 16 neurons. Most models appear to have the same issue as the linear model, with some deviation from the expected linear increase in fouling during the second fouling period. This can be seen around timestamp 2020.06 in the figure. Notice that the more complex MLP models reset to a more sensible level than the linear model at the end of the time series, predicting a slightly negative deviation rather than a large positive deviation.

Figure C.5c shows the deviation between predicted and measured value for an ensemble model consisting of an MLP model using a single hidden layer of 128 neurons and a linear regression model. The achieved deviation pattern is very similar to the expected fouling pattern. In general, the use of ensemble models is found to improve the performance very slightly for all facilities, although without significant increases in fouling estimability.

Figure C.4 shows predictions and measurements for facility G, using models with 64 and 128 neurons. Notice that some of the models converge to slightly different solutions. This is particularly visible between timestamp 2018.01 and 2018.05. This deviation is quite similar to what was found for different levels of regularization. It is clear that using different architectures and regularization parameters make the model converge to different parts of the solution space.

To conclude, models of one hidden layer with 128 neurons, or models with two hidden layers and either 64 or 128 neurons, are found to achieve the highest model performance. More complex architectures are not tested due to the rather high dataset sampling rates and lack of appropriate hardware. As a default, one hidden layer with 128 neuron is chosen due to acceptable performance and relatively short convergence time. For data with higher sampling rates, increasing the network depth and width may be applicable.

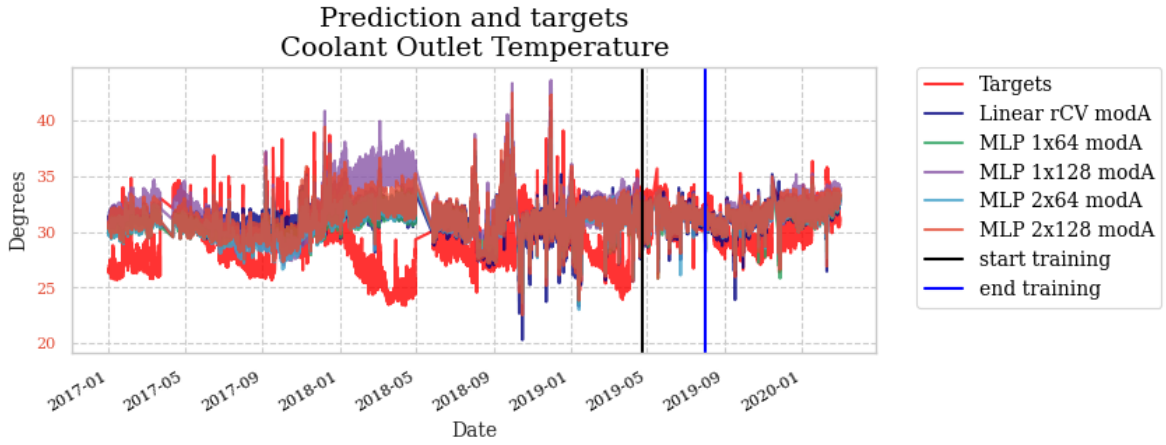
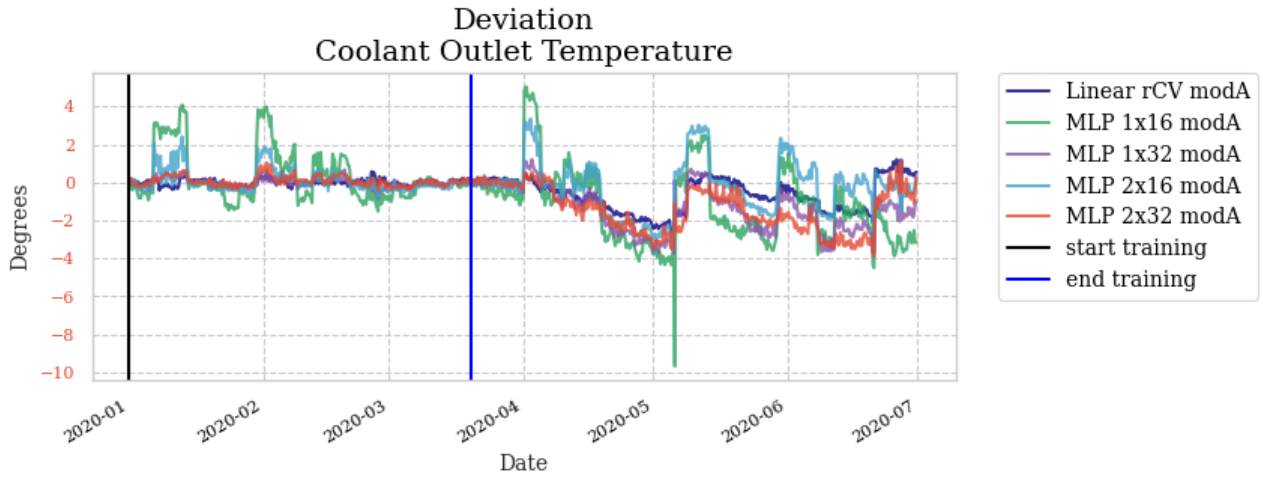
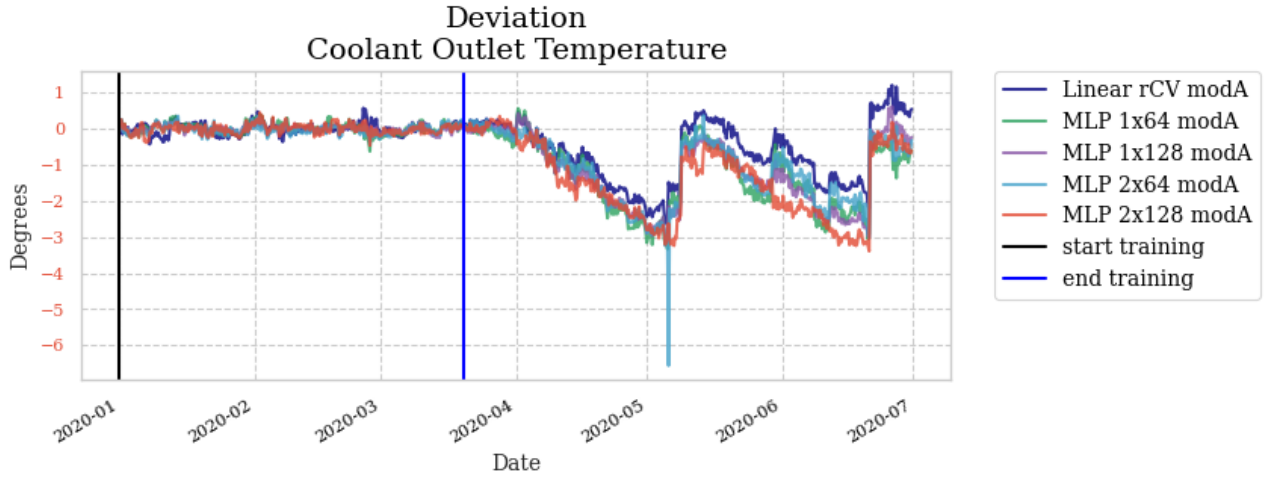


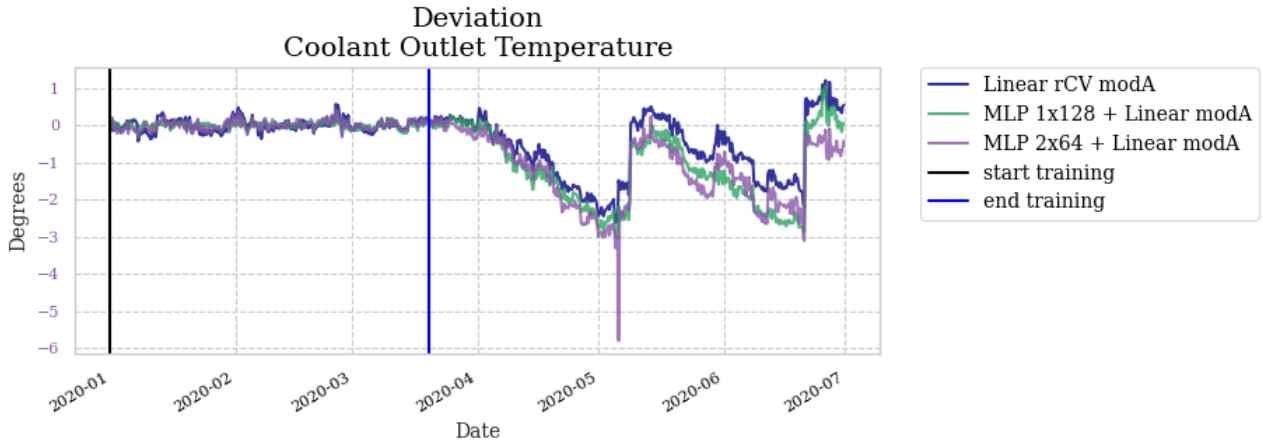
Figure C.4: Predicted values for models with 64 and 128 neurons, dataset G.



(a) Deviation between predicted and measured value for models with 16 and 32 neurons, dataset D.



(b) Deviation between predicted and measured value for models with 64 and 128 neurons, dataset D.



(c) Deviation between predicted and measured value for ensemble models, dataset D.

Figure C.5: MLP architecture comparison for dataset D.

C.3 Recurrent neural networks

C.3.1 LSTM dropout regularization

Dataset G is used with sampling rates of 30 and 10 minutes and predictive model A. The architecture is chosen as one hidden layer of 128 neurons for each model to ensure sufficient model capacity with acceptable training time. The same level of regular dropout and recurrent dropout is added for each model. Calculated metrics are seen in Table C.9. Be reminded that the loss metric provided in Table C.9 is the chosen loss corresponding to the epoch with minimum validation loss, not the minimum loss. Additional results and plots, including the minimal loss, can be seen on GitHub[5].

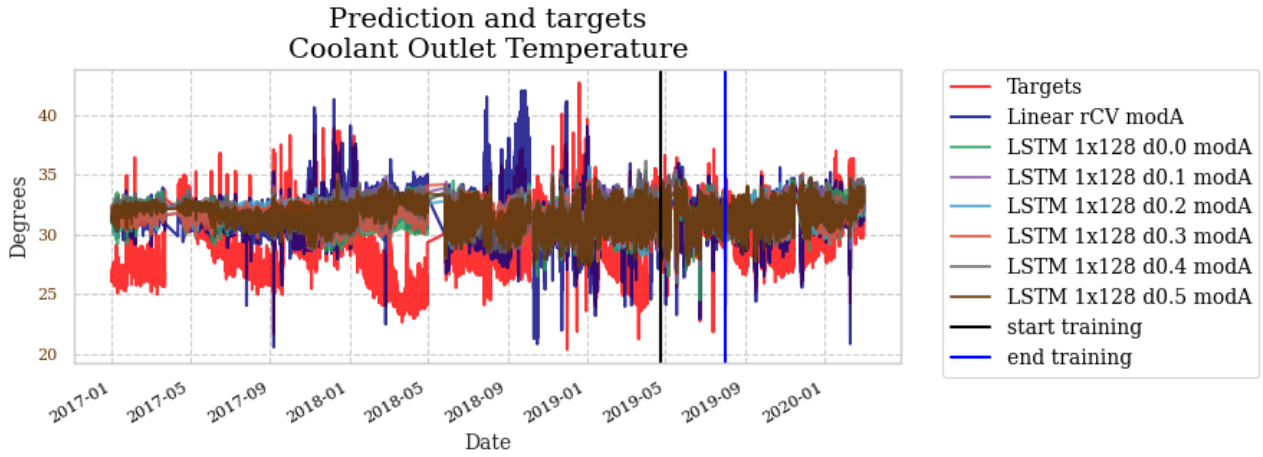
Comparable performance is found for different dropout rates using both sampling rates. In fact, the different rates are found to converge to very similar solutions. The calculated validation loss is generally low, however particularly so for the model using a dropout rate of 0.2. Somewhat surprisingly, the model using no dropout only obtains a slightly lower minimum loss than the remaining models. This indicates that the model is not able to overfit the data to any extended degree despite lack of regularization, at least within the limited number of epochs allowed before early stopping aborts the training process. However, this does not necessarily mean that the model would not overfit further given additional training epochs.

Figures C.6a and C.6b show the calculated predictions and deviations. Some notable differences are observed around timestamp 2018.05. The models otherwise predict roughly the same values with the same degree of variation. Predictions in general have fewer large variations than the linear model. It is concluded that a dropout rate of 0.2 is preferred, as this ensures the best compromise between model performance and speed of convergence.

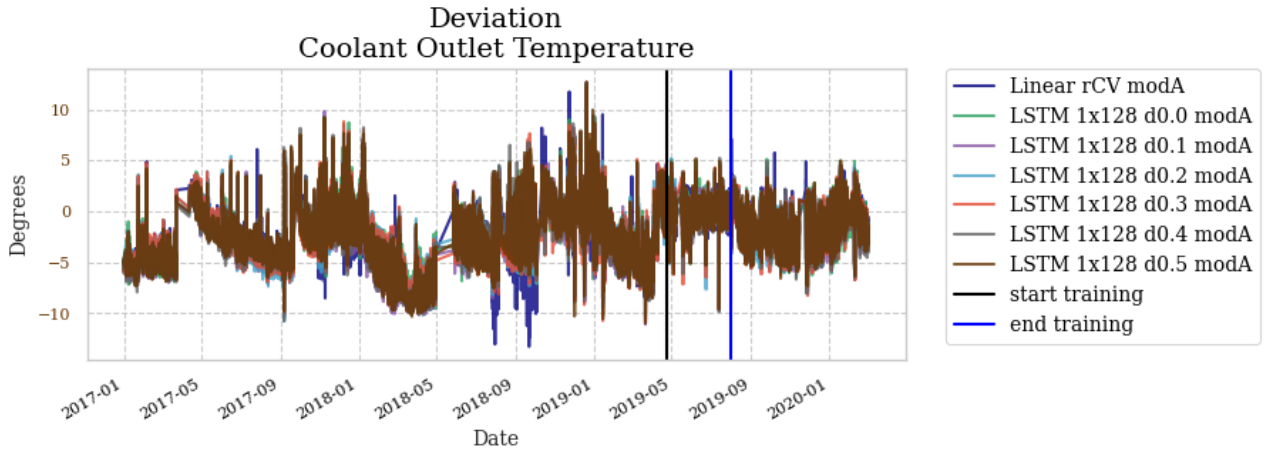
Table C.9: LSTM dropout comparison metrics for dataset G

	30min sampling rate			10min sampling rate		
Dropout	Loss	Val.loss	R^2	Loss	Val.loss	R^2
0.0	0.2068	0.3008	0.8732	0.2137	0.3363	0.8600
0.1	0.1774	0.3226	0.8860	0.2431	0.3451	0.8378
0.2	0.2145	0.2298	0.8797	0.2462	0.3380	0.8278
0.3	0.2215	0.3066	0.8615	0.2467	0.3522	0.8264
0.4	0.2082	0.3390	0.8582	0.2583	0.3481	0.8244
0.5	0.2374	0.3227	0.8498	0.2613	0.3412	0.8242

[5] https://github.com/hermanwh/master-thesis/blob/master/3_lstm_dout_comp.ipynb



(a) Predicted and measured values for dataset G, 10min sampling rate.



(b) Deviation between predicted and measured values for dataset G, 10min sampling rate.

Figure C.6: MLP dropout comparison.

C.3.2 LSTM architecture

As for the MLP architectures presented in section C.2.3, this chapter demonstrates different architectures for LSTM networks. In LSTM networks, the lookback parameter is an additional hyperparameter which may be varied. For these comparisons, a lookback parameter ensuring a lookback duration of 6 hours is used. Datasets F and G are used with a sampling rate of 30 minutes. Note that for large datasets with sizeable lookback parameters, the training of recurrent neural networks can be very time consuming, especially without powerful hardware. This limits the number of applicable architectures. Predictive models A and B are both tested. Dropout regularization with a rate of 0.2 is used for all models. Recurrent dropout with the same rate is also added.

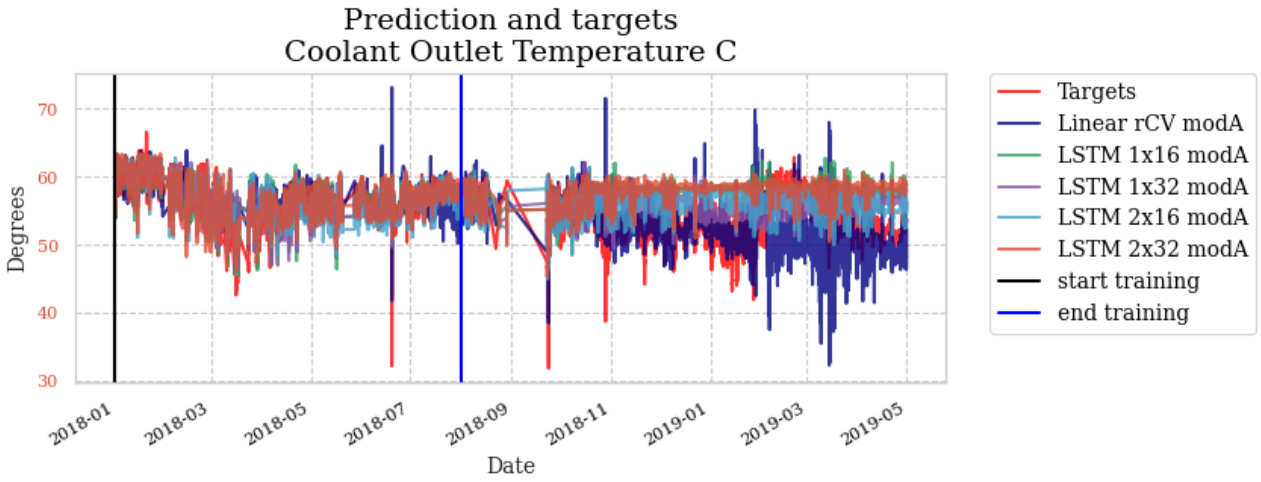
Calculated metrics are seen in Table C.10. Additional results and plots can be seen on GitHub[6]. Perhaps most interestingly, the predictions for dataset F appear to have much less variation for complex model architectures, as seen in figures C.7a, C.7b. Particularly the model with two hidden layers of 128 neurons predict almost a strict line, with much less variation than what is observed for models with 16 and 32 neurons. It is difficult to determine which model produces the most sensible predictions, even when simultaneously inspecting the dataset feature plots. Figure C.7c shows the models with 64 and 128 neurons for facility G, for which this same lack of variation is not observed. More research is necessary to interpret these results. Notice that the LSTM models for dataset G with predictive model B do not overfit to the valve opening feature, like the linear model does.

Loss metrics are generally lower for the models with higher width, with some exceptions. Interestingly, the validation loss for predictive model B is only slightly lower than that of predictive model A. However, a significant increase in R^2 is observed. To conclude, an architecture of one hidden layer with 128 neurons is preferred, as was concluded for MLP architectures. The reasoning is that this model shows some more variation than the suspect complex models discussed earlier, while maintaining acceptable performance and convergence time.

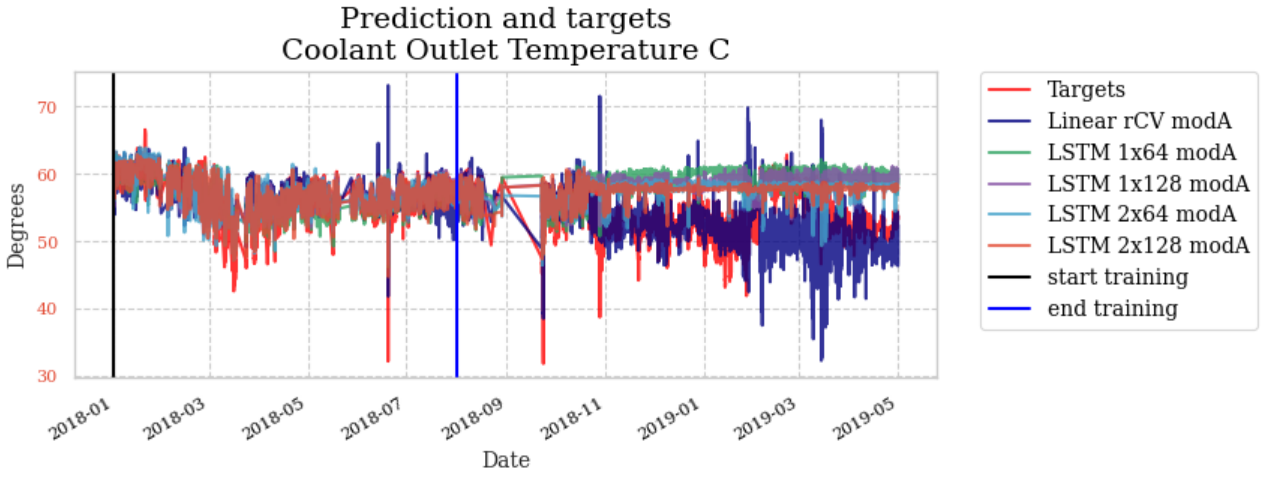
Table C.10: LSTM architecture comparison metrics, datasets F and G

		Model A			Model B		
Dataset	Layers	Loss	Val.loss	R^2	Loss	Val.loss	R^2
F	1x 16	0.1389	0.1911	0.5872	0.1492	0.1665	0.6826
F	1x 32	0.1065	0.1431	0.5922	0.0888	0.1331	0.7582
F	2x 16	0.1043	0.1625	0.5176	0.1161	0.1423	0.8301
F	2x 32	0.0589	0.1209	0.5065	0.0641	0.1134	0.7887
F	1x 64	0.0579	0.1271	0.5766	0.0833	0.1091	0.7916
F	1x128	0.0468	0.1181	0.5912	0.0517	0.1191	0.7840
F	2x 64	0.0474	0.1282	0.5955	0.0500	0.1079	0.8233
F	2x128	0.0375	0.1124	0.6553	0.0382	0.1160	0.7984
G	1x 16	0.2567	0.3253	0.6423	0.2927	0.3075	0.7763
G	1x 32	0.2137	0.2630	0.5780	0.2536	0.2744	0.7766
G	2x 16	0.2443	0.2995	0.5423	0.2487	0.2615	0.7572
G	2x 32	0.2181	0.3032	0.4840	0.1826	0.2574	0.7638
G	1x 64	0.1976	0.3094	0.6305	0.2492	0.2860	0.7829
G	1x128	0.2476	0.2857	0.6568	0.2077	0.2658	0.7898
G	2x 64	0.2098	0.2833	0.6065	0.2127	0.2721	0.7844
G	2x128	0.1582	0.2737	0.6113	0.1847	0.2671	0.7849

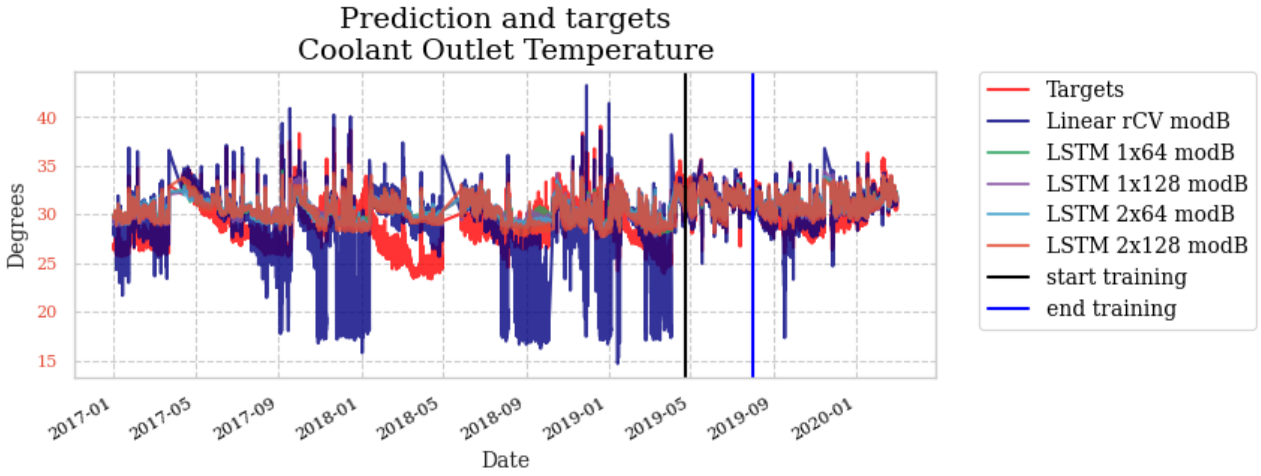
[6] https://github.com/hermanwh/master-thesis/blob/master/3_lstm_arch_comp.ipynb



(a) Predicted and measured values for models with 16 and 32 neurons.



(b) Predicted and measured values for models with 64 and 128 neurons.



(c) Predicted and measured values for models with 64 and 128 neurons.

Figure C.7: LSTM architecture comparison for dataset F and G.

C.4 Predicting with model uncertainty

When dropout regularization in neural networks is enabled at the time of prediction, each prediction is inherently stochastic. By obtaining a sufficient number of predictions, the predictive distribution of the trained model may be obtained. This distribution can be used to assess the model uncertainty. For example, if the predictive distribution of the model is obtained while also assessing the underlying data uncertainty, a predictive distribution may be obtained which can be used in various predictive maintenance schemes, such as estimation of remaining useful life. It is important to note that assessing the model uncertainty alone is not sufficient to establish a model for the total uncertainty. To do so, the underlying uncertainty related to the acquired data must also be assessed, which is not achieved through these experiments.

In some systems, such as in batch production systems, the lookback duration of recurrent neural networks may easily be determined. However, for heat exchangers there is no natural lookback duration. The lookback parameter is chosen so that a lookback duration of 6 hours is obtained, meaning a lookback parameter of 12 for a sampling rate of 30 minutes and 36 for a sampling rate of 10 minutes. It is expected that increasing the number of data samples, and thus the lookback duration, will reduce the uncertainty of the trained models.

Because recurrent networks were found to have limited predictive capabilities for facility D, LSTM and GRU models are fitted for facilities F and G. For facility F, a sampling rate of 30 minutes is used. For facility G, sampling rates of both 30 and 10 minutes are used to investigate if increased sampling rate reduces model uncertainty. The chosen datasets contain 8222, 4232 and 12669 samples, respectively. The resulting training loss and validation loss values are seen in Table C.11. 20% validation data is chosen at random from the training data. LSTM models are found to have better performance than GRU models. Additional results and plots can be seen on GitHub[7]

Figures C.9a, C.9b and C.9c show the predictive performance for each of the three datasets. The means and standard deviations of 30 individual predictions are used. The mean value is shown in blue, while the target value is shown in red. Bounds obtained by adding and subtracting the standard deviation of the predictions are

[7] https://github.com/hermanwh/master-thesis/blob/master/4_pred_uncertainty.ipynb

Table C.11: Predicting with model uncertainty metrics, datasets F and G

	F 30min		G 30min		G 10min	
Model	Loss	Val.loss	Loss	Val.loss	Loss	Val.loss
LSTM 1x128	0.2906	0.3025	0.3001	0.3526	0.3116	0.3709
GRU 1x128	0.3273	0.3395	0.3778	0.3919	0.3370	0.3719

shown in gray. The model uncertainty is rather small during the training period, while somewhat larger during the testing period, as is expected. Even when accounting for the uncertainty bounds, the curves for predicted and targeted values are easily separable with regards to long-term variations, indicating that the predictive models are suitable for determining the deviation between predicted and measured value. Figures C.10a, C.10b and C.10c show the convergence curves for the trained models, all of which appear reasonable.

As indicated by figures C.9b and C.9c, the sampling rate appears to have some, although limited, effect on the uncertainty bounds for the obtained datasets. GRU models are found to produce predictions with higher degree of variation than LSTM models, given the same lookback duration. Additionally, the uncertainty appears greater, indicated by a larger standard deviation between predictions. An example for facility F can be seen in Figure C.8. When the lookback duration is increased, GRU model predictions seem to diverge drastically. Overall, no obvious reasons for choosing GRU in favor of LSTM are found.

Note that uncertainty analysis is not an objective of this thesis, even though this is fundamental in the development and usage of any predictive model in practice. The means of analysing model uncertainty for models using dropout are presented primarily to demonstrate the capabilities of the implemented solution and to encourage further work on uncertainty assessment. Recommendations for such work are discussed further in 9.3.

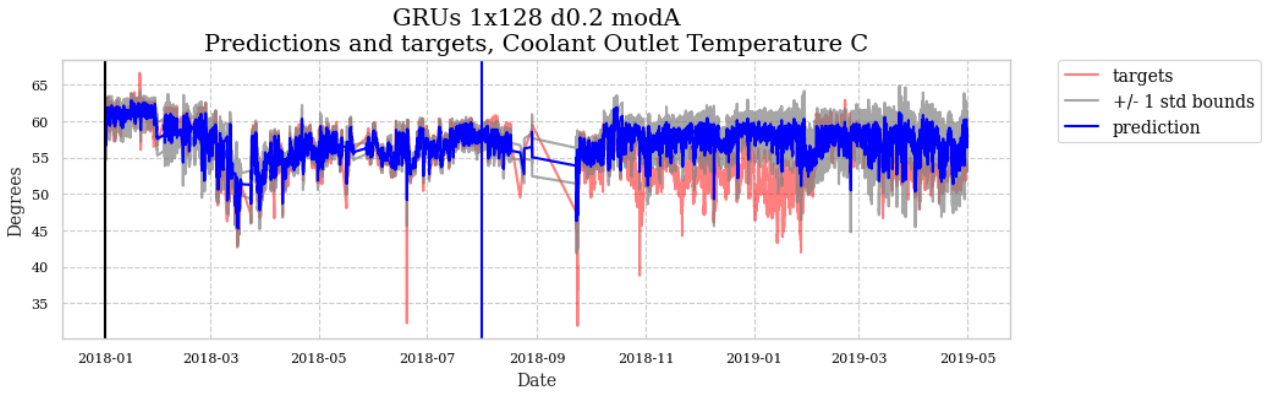
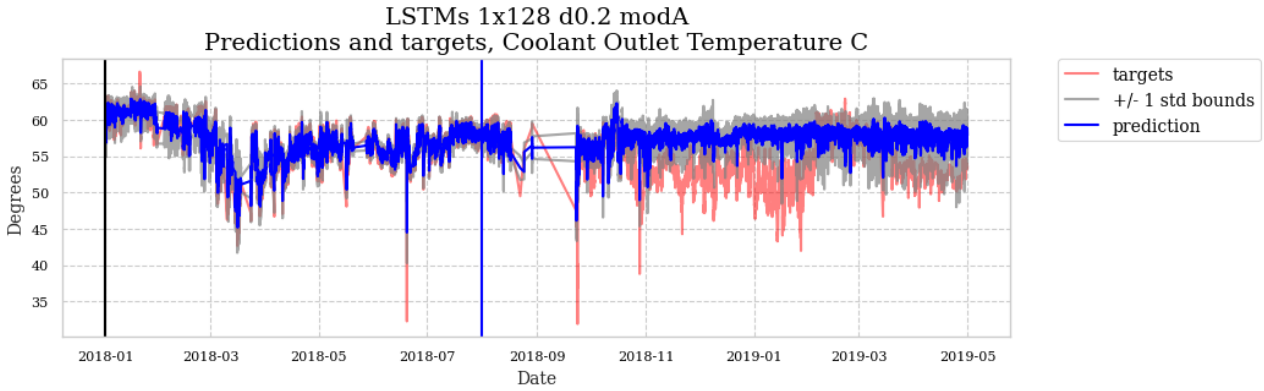
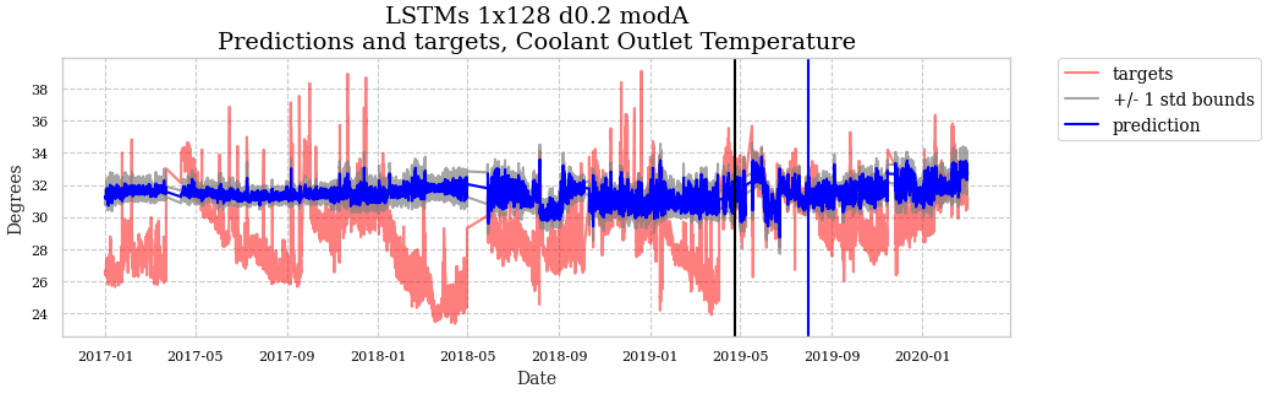


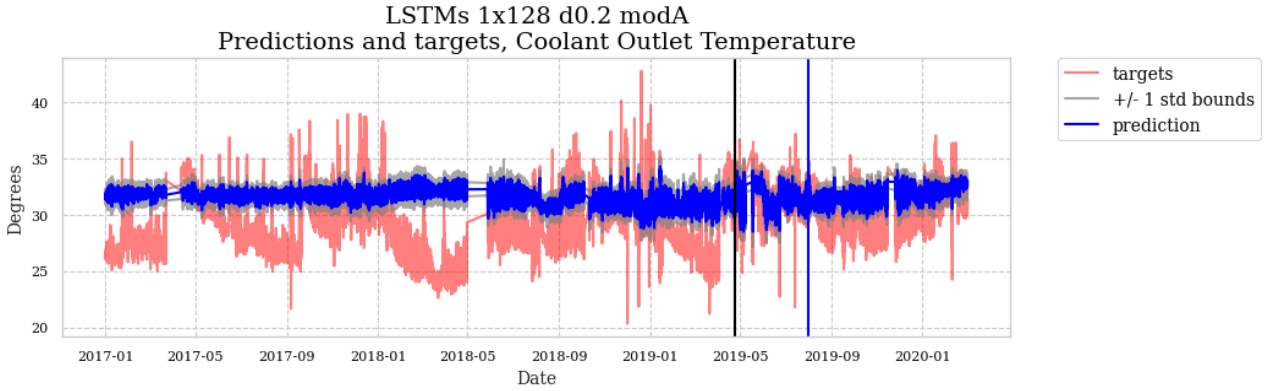
Figure C.8: Predictions for facility F with sampling rate of 30 minutes using GRU model with dropout at time of prediction.



(a) Facility F with sampling rate of 30 minutes.

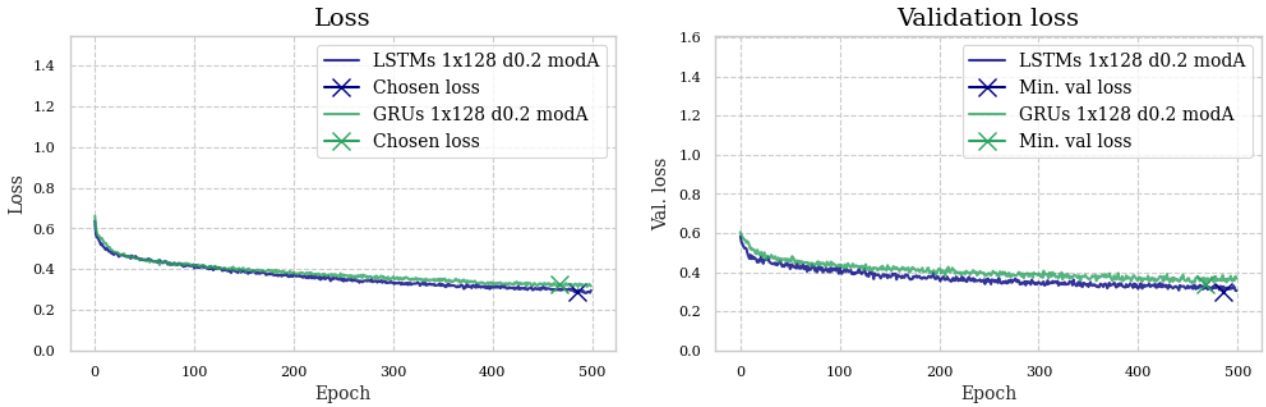


(b) Facility G with sampling rate of 30 minutes.

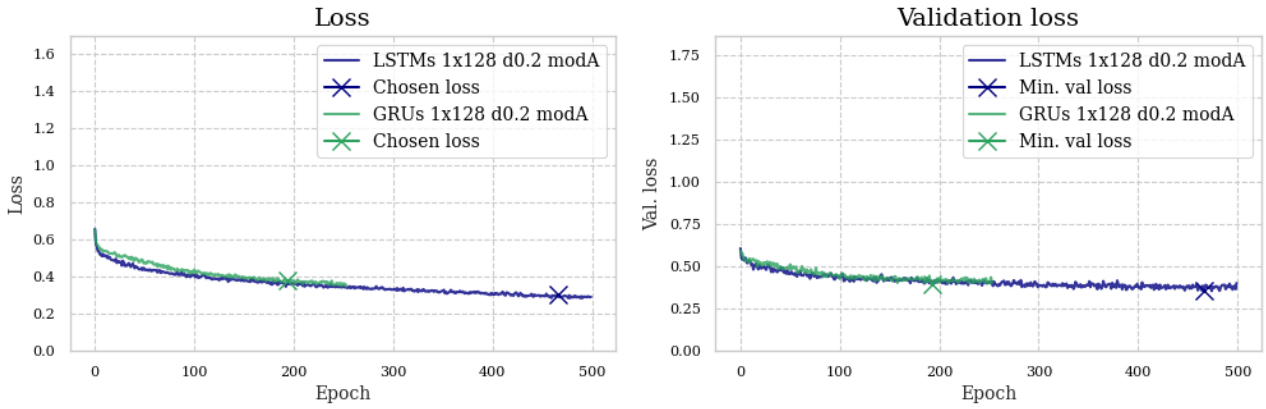


(c) Facility G with sampling rate of 10 minutes.

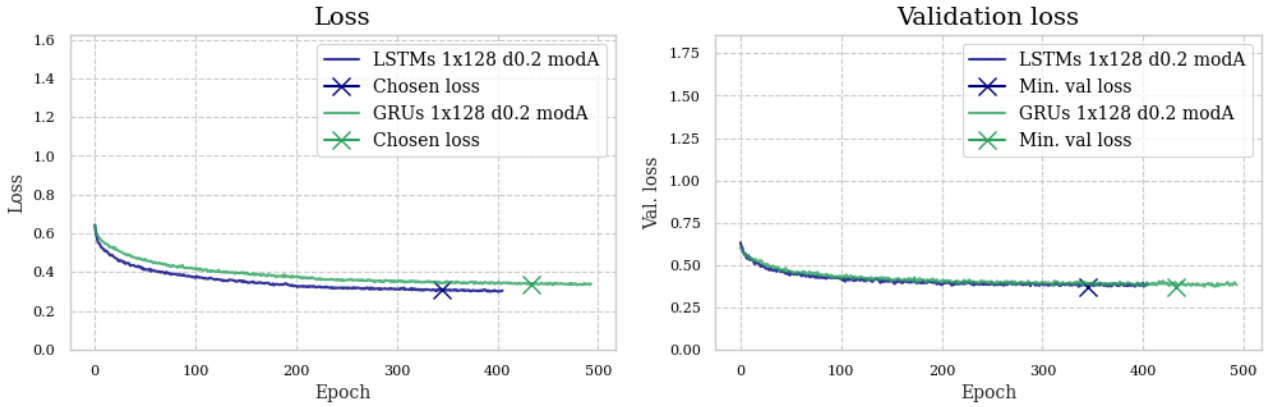
Figure C.9: Predicted and measured values using LSTM model with dropout at time of prediction.



(a) Facility F with sampling rate of 30 minutes.



(b) Facility G with sampling rate of 30 minutes.



(c) Facility G with sampling rate of 10 minutes.

Figure C.10: Convergence pattern for models using LSTM model with dropout at time of prediction.

Appendix D

Additional results

D.1 Unsupervised methods

D.1.1 Principal component analysis

D.1.2 Correlation plots

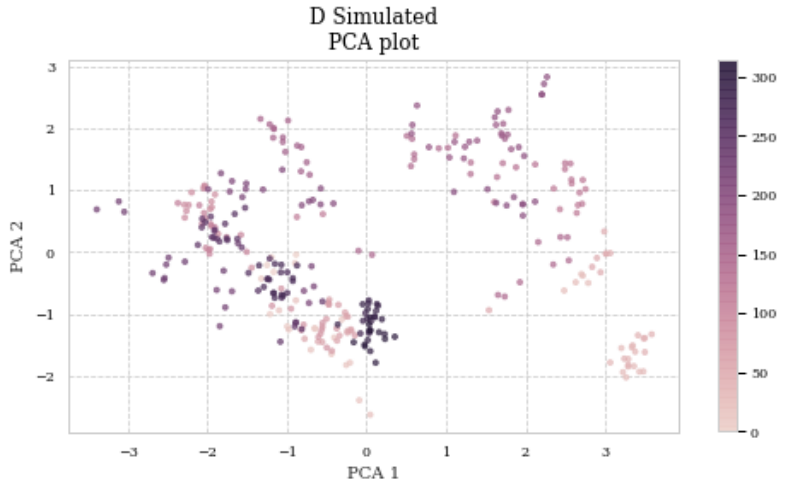
D.2 Supervised methods

D.2.1 Facility D

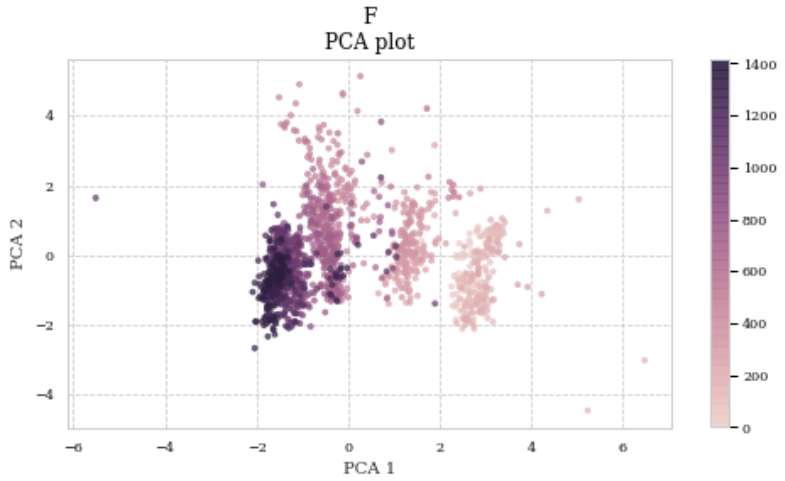
D.2.2 Facility F

D.2.3 Facility G

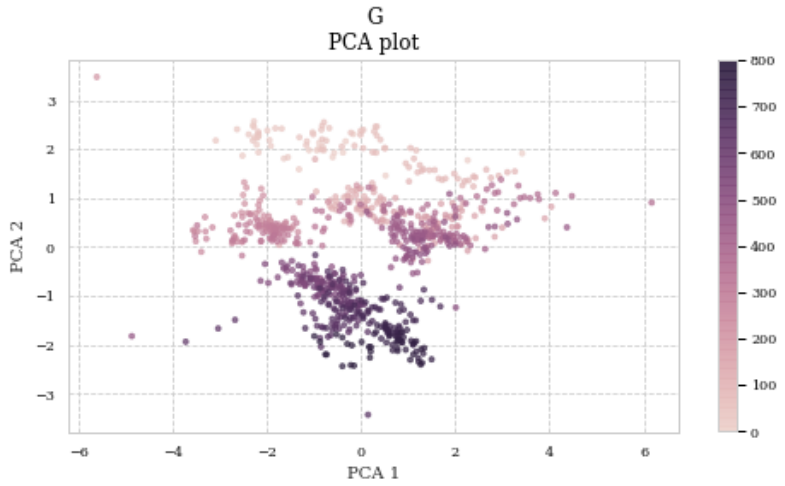
D.3 Predicting across facilities



(a) Principal component analysis of dataset D training data.

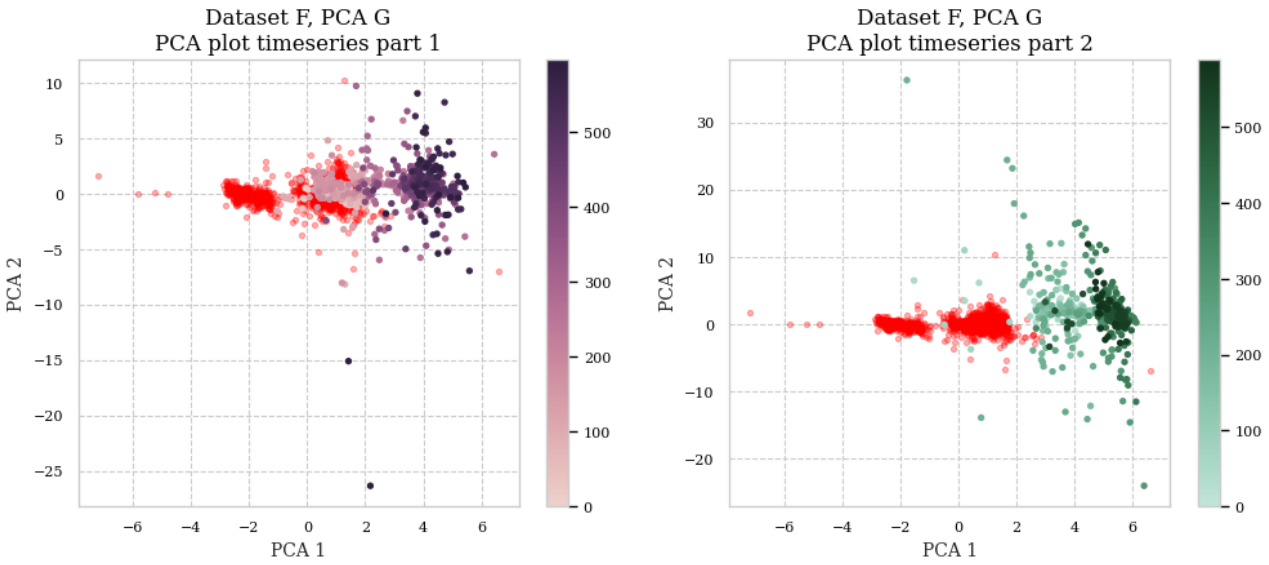


(b) Principal component analysis of dataset D training data.

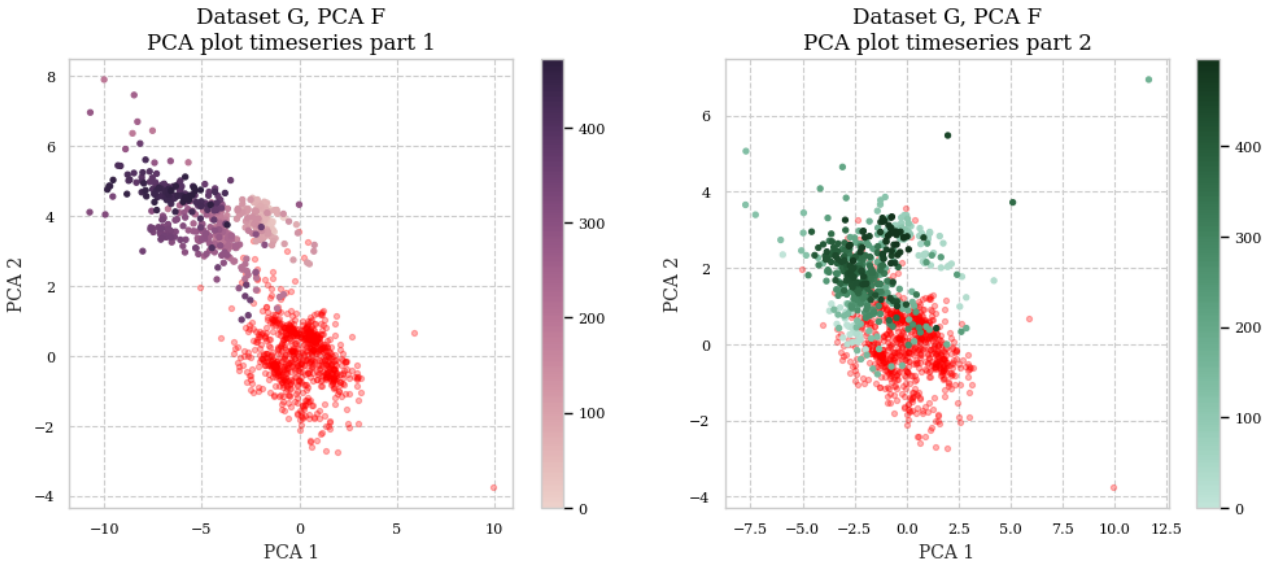


(c) Principal component analysis of dataset D training data.

Figure D.1

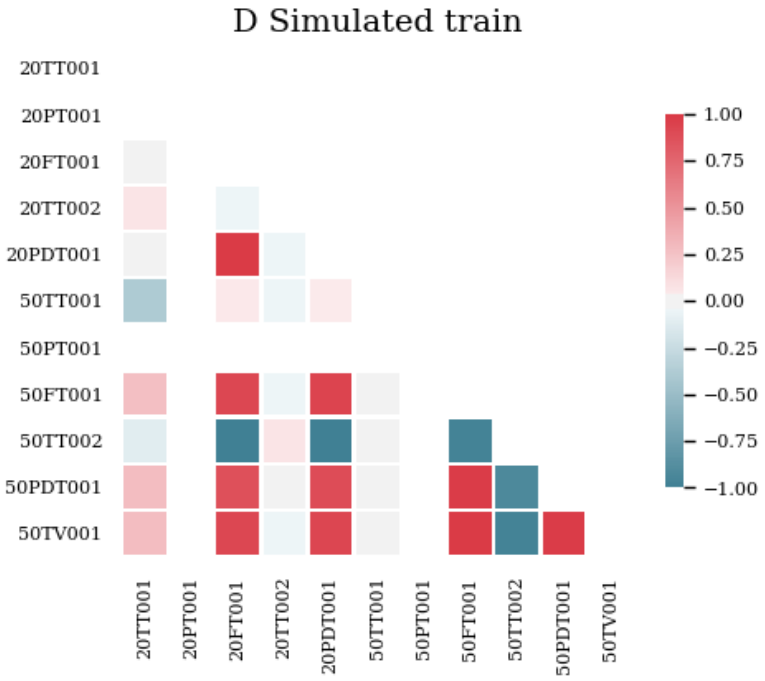


(a) Principal component analysis of dataset F with PCA model fitted to dataset G.

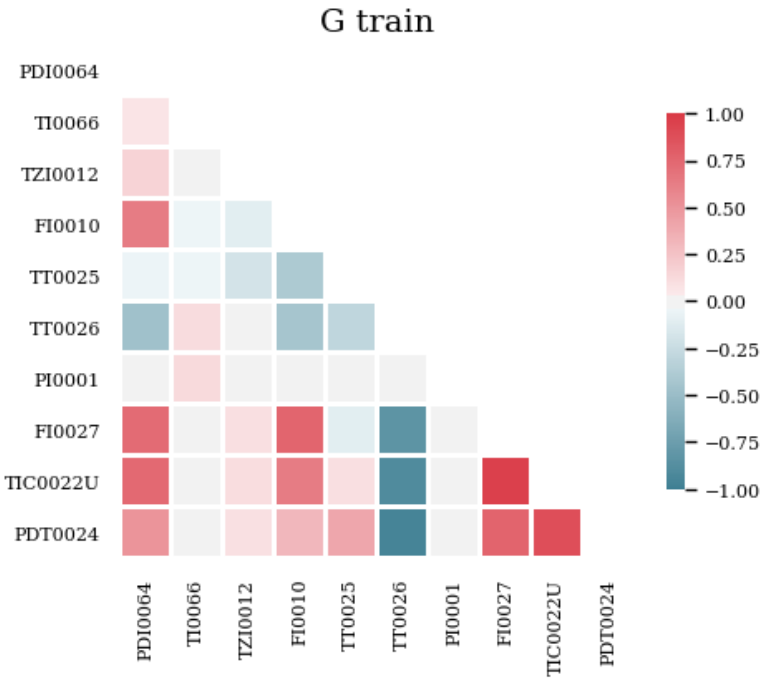


(b) Principal component analysis of dataset G with PCA model fitted to dataset F.

Figure D.2

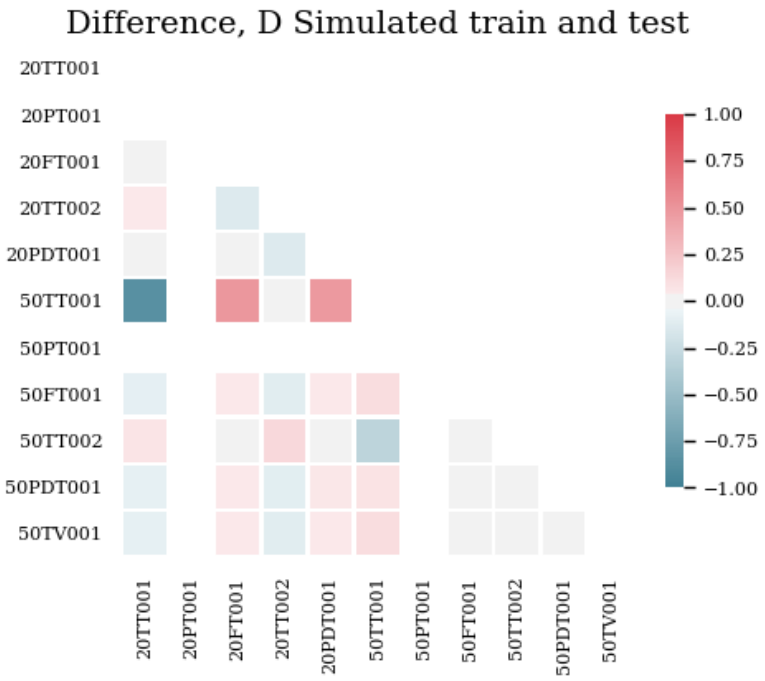


(a) Correlation matrix for dataset D.

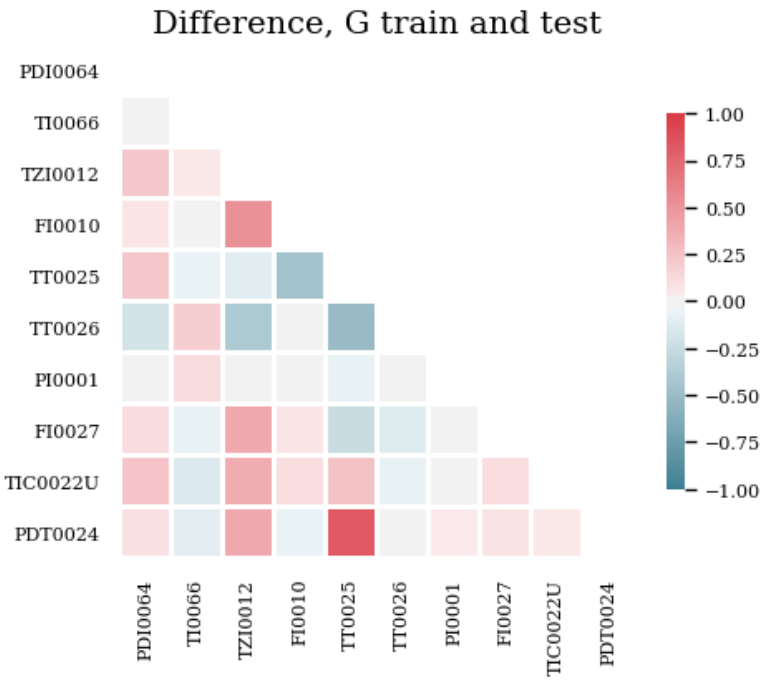


(b) Correlation matrix for dataset G.

Figure D.3

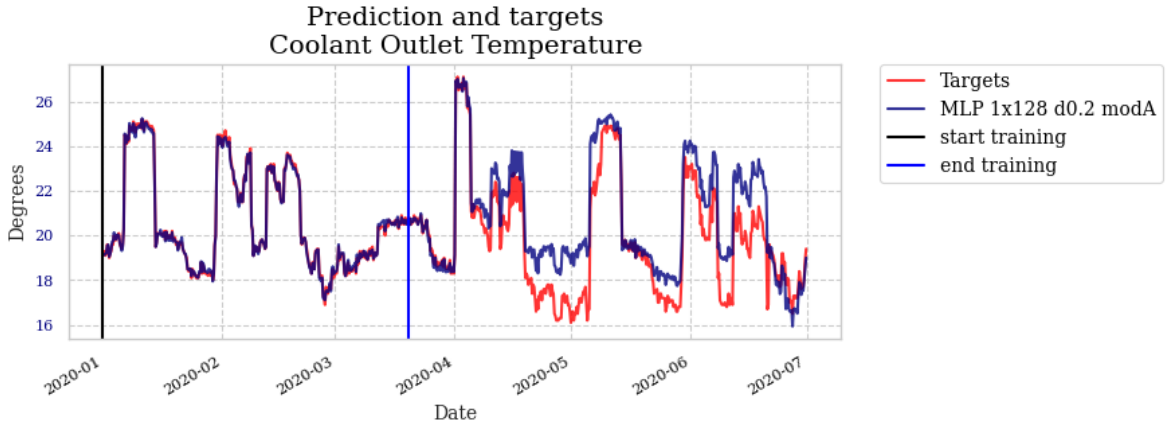


(a) Difference between the correlation matrices during training and testing for dataset D.

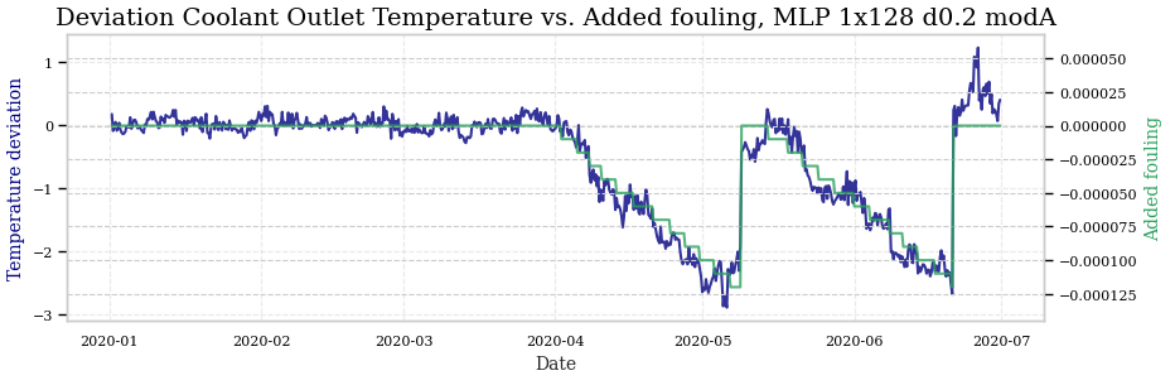


(b) Difference between the correlation matrices during training and testing for dataset G.

Figure D.4

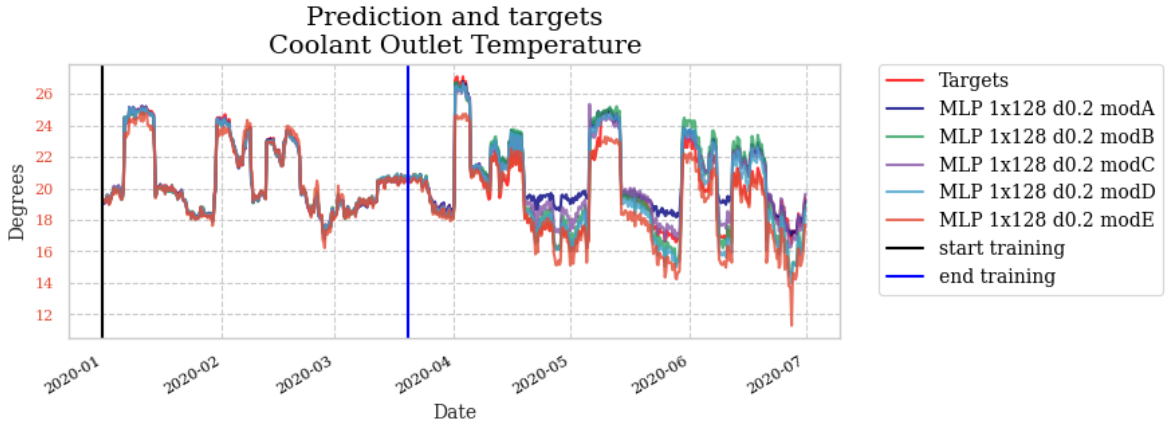


(a) Predictions and measured values.

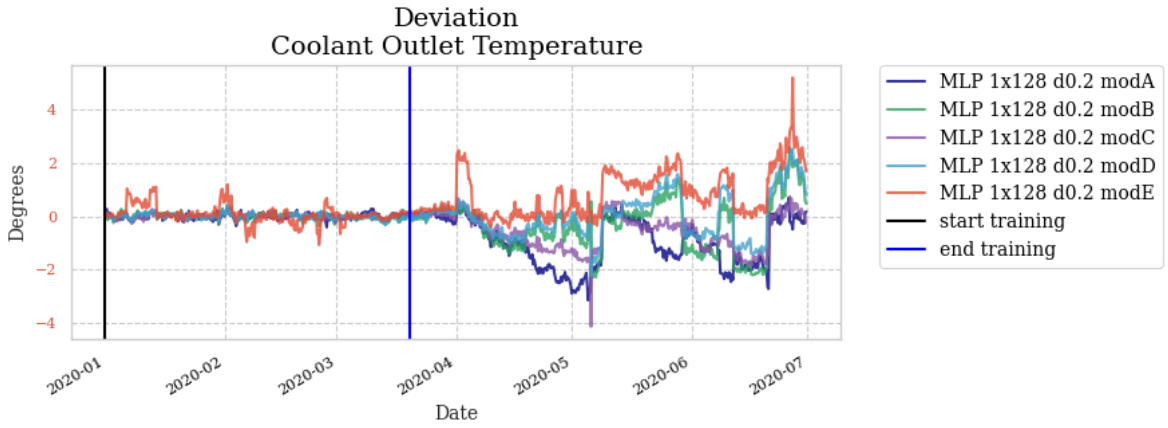


(b) Deviation between predicted and measured values on the left axis. Added fouling on the right axis.

Figure D.5: Predictions and deviations for $T_{c,o}$ on dataset D using a MLP model.

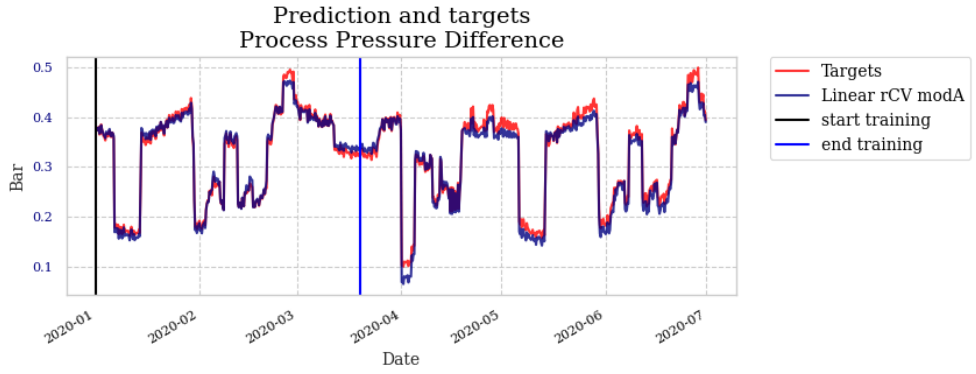


(a) Predictions and measured values.

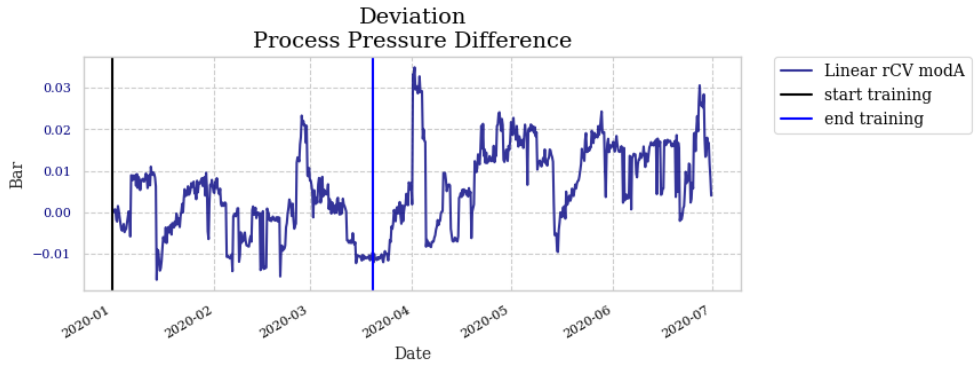


(b) Deviation between predicted and measured values.

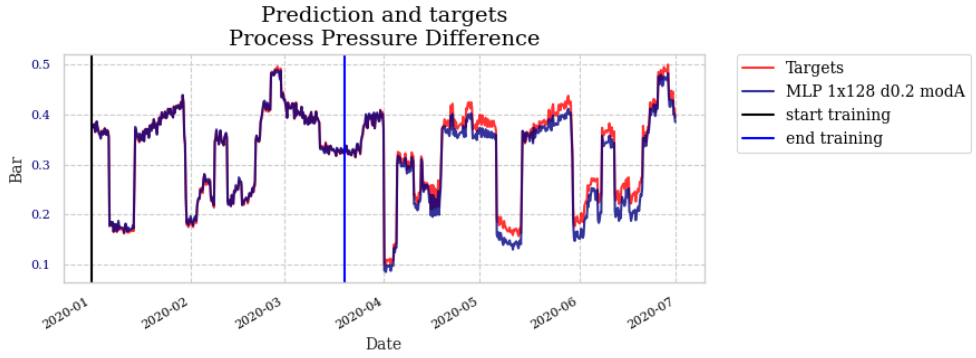
Figure D.6: Predictions and deviations for $T_{c,o}$ on dataset D using predictive models A to E.



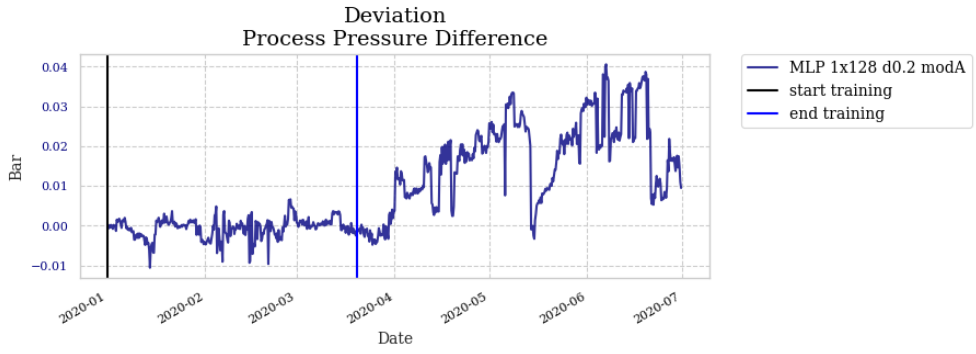
(a) Predictions and measured values for dP_h, linear regression model.



(b) Deviation between predicted and measured dP_h, linear regression model.

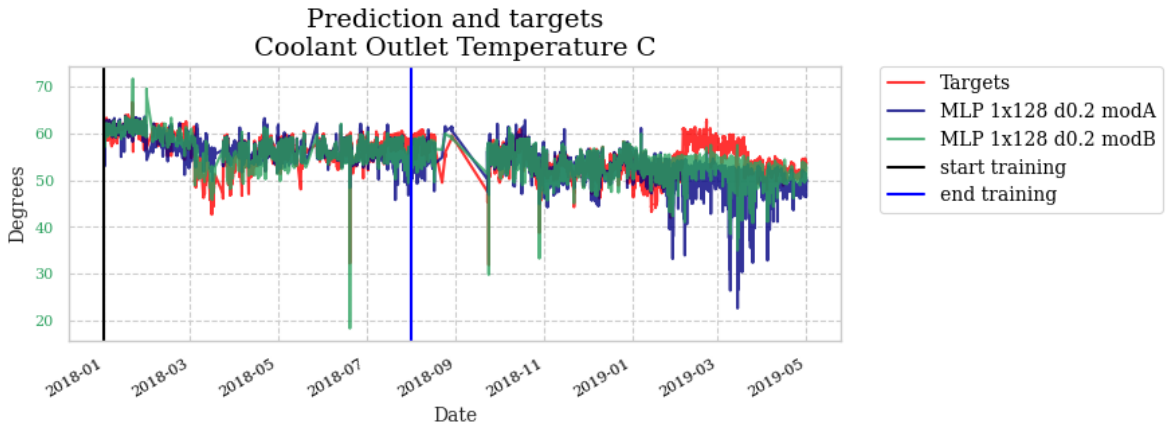


(c) Predictions and measured values for dP_h, MLP model.

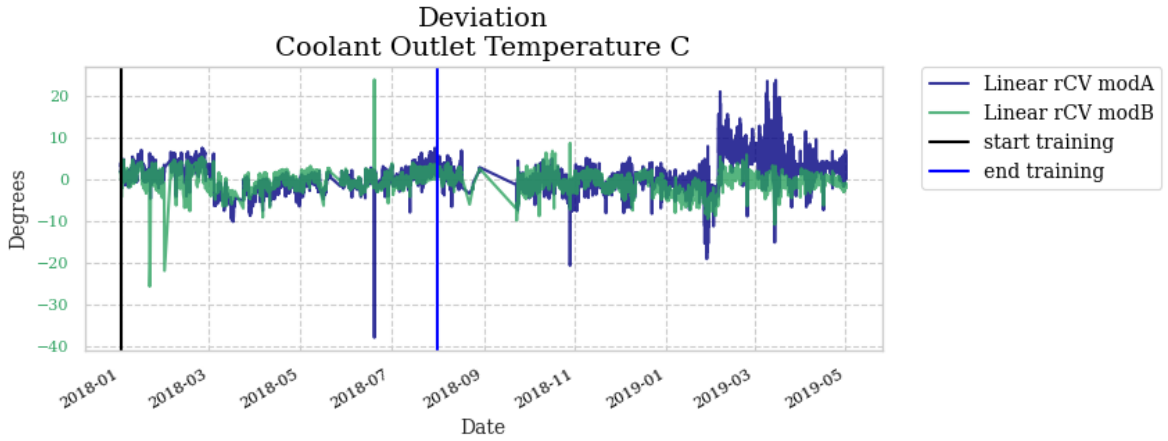


(d) Deviation between predicted and measured dP_h, MLP model.

Figure D.7: Predictions and deviations of dP_h for dataset D with fouling on the process side.

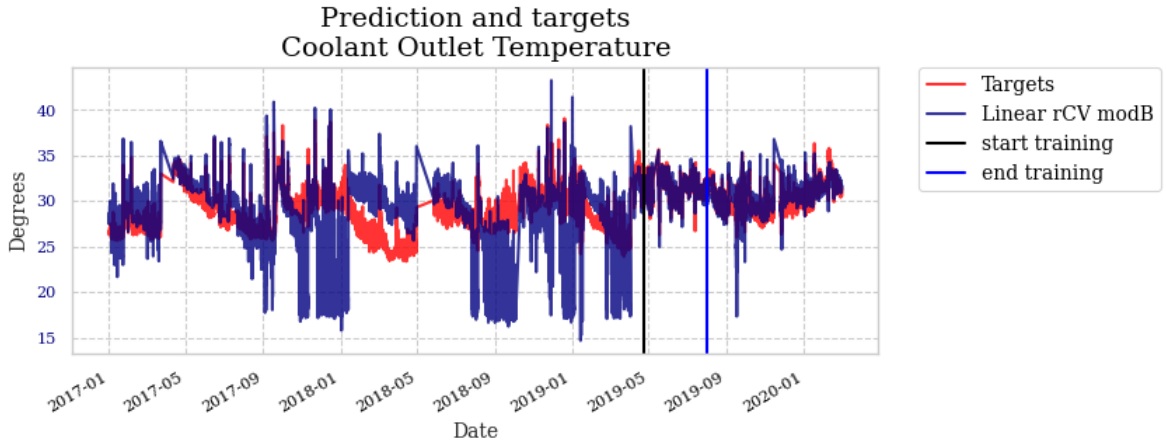


(a) Predictions and measured values.

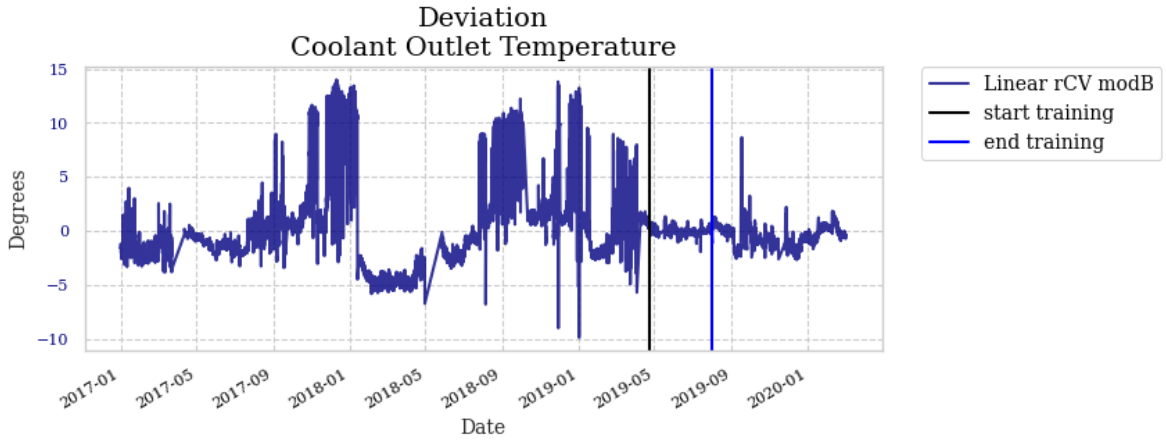


(b) Deviation between predicted and measured values.

Figure D.8: Predictions and deviations for $T_{c,o}$ on dataset F using MLP models.

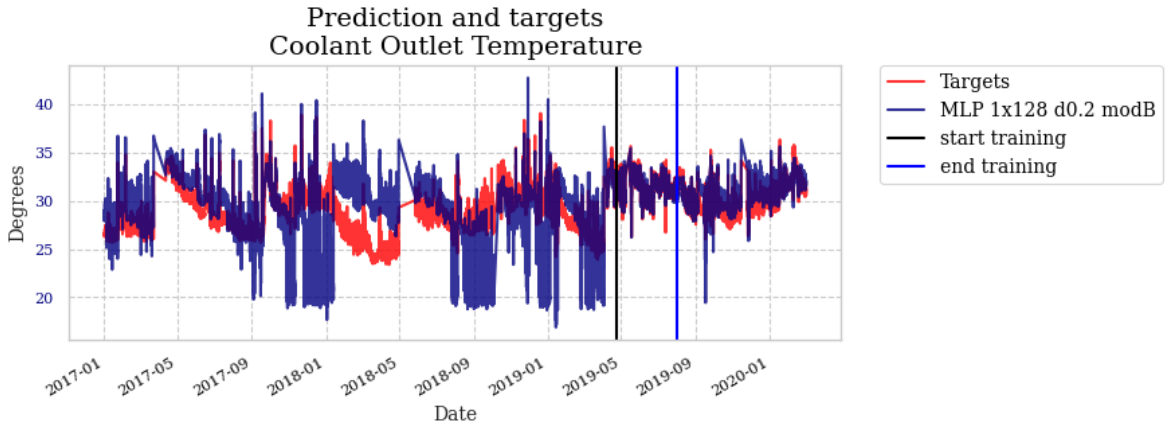


(a) Predictions and measured values.

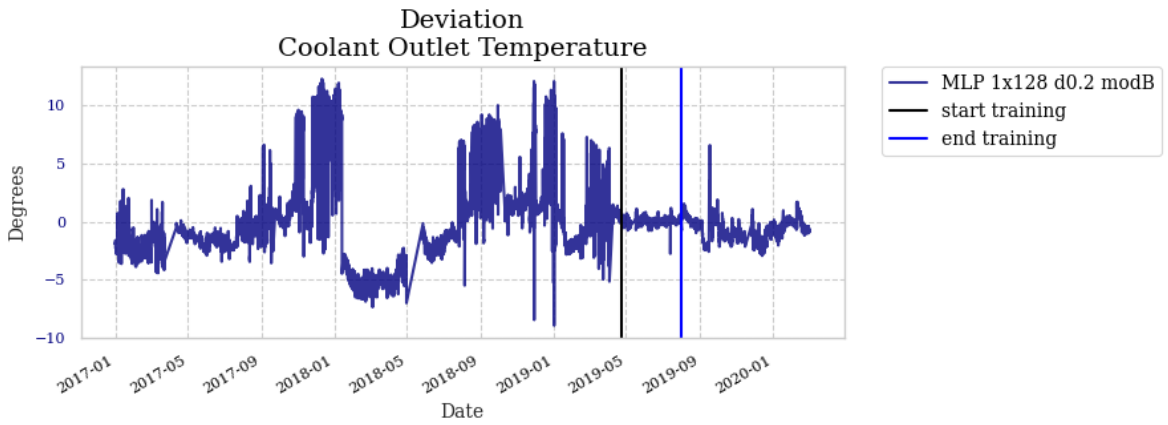


(b) Deviation between predicted and measured values.

Figure D.9: Predictions and deviations for $T_{c,o}$ on dataset G using linear models and predictive model B.

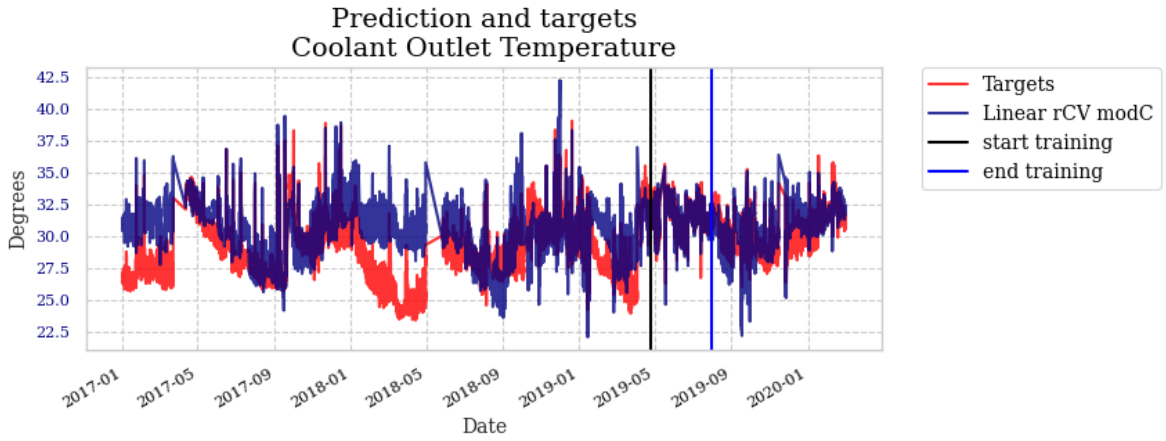


(a) Predictions and measured values.

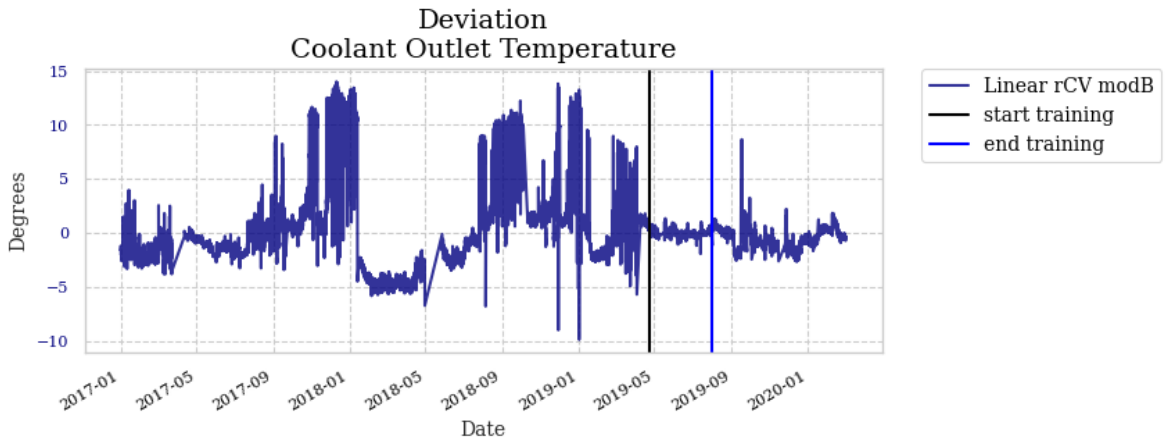


(b) Deviation between predicted and measured values.

Figure D.10: Predictions and deviations for $T_{c,o}$ on dataset G using MLP models and predictive model B.

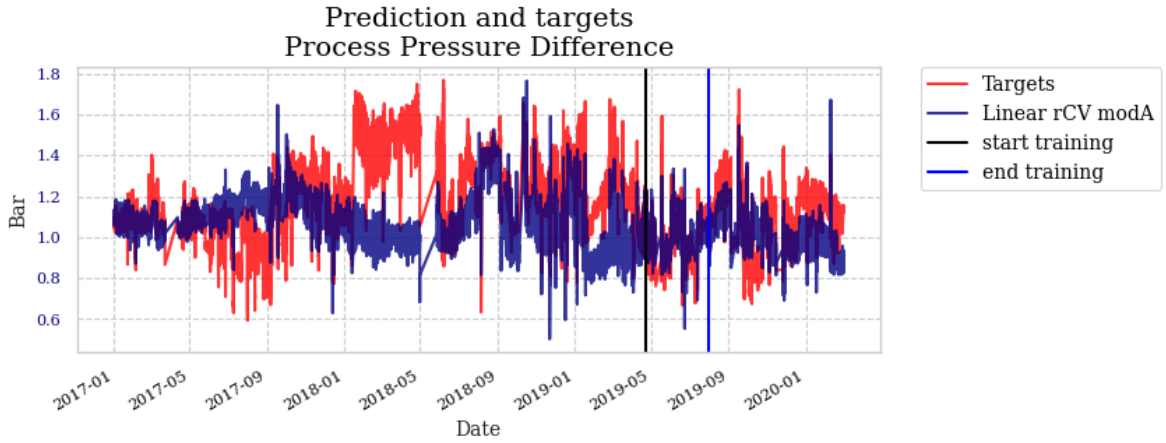


(a) Predictions and measured values.

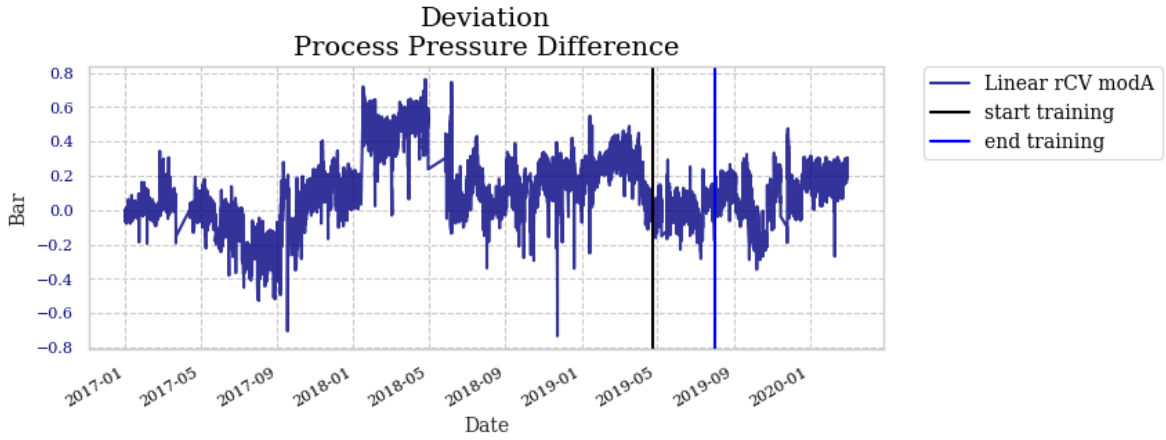


(b) Deviation between predicted and measured values.

Figure D.11: Predictions and deviations for $T_{c,o}$ on dataset G using linear models and predictive model B and C.

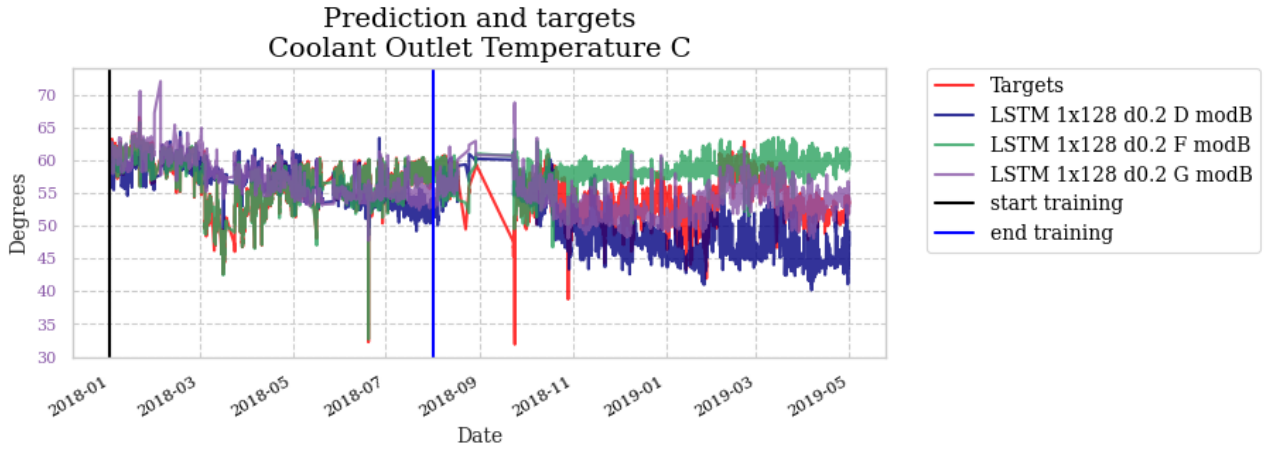


(a) Predictions and measured values.

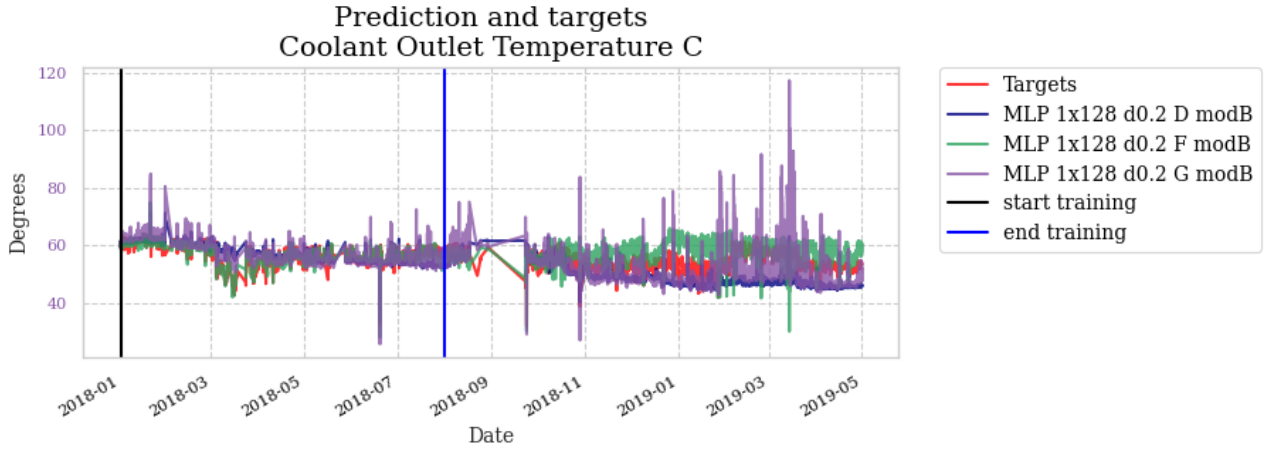


(b) Deviation between predicted and measured values.

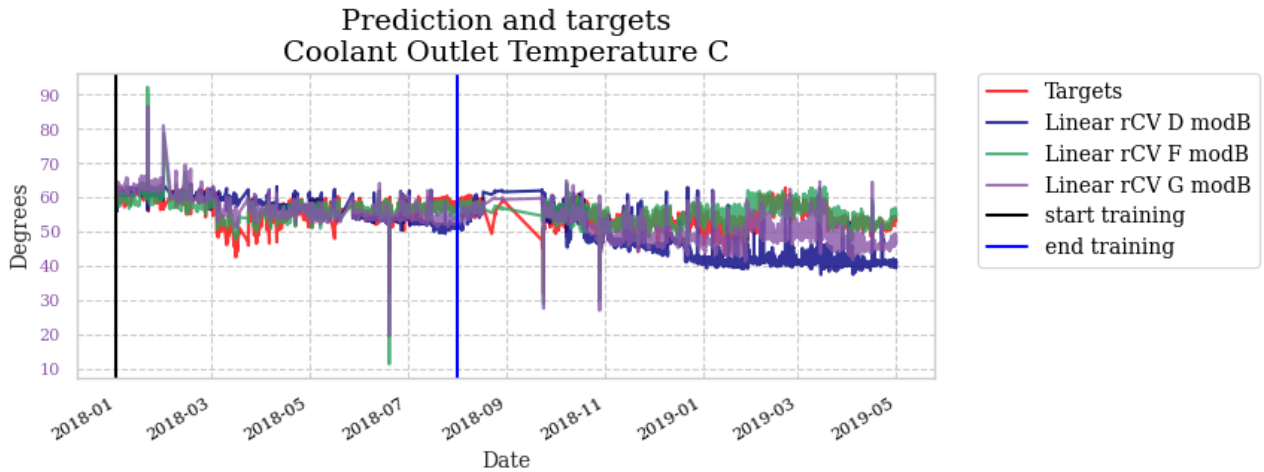
Figure D.12: Predictions and deviations for dP_h on dataset G using linear models and predictive model A.



(a) Predictions for facility F using LSTM models trained on facilities D, F and G.



(b) Predictions for facility F using MLP models trained on facilities D, F and G.



(c) Predictions for facility F using linear models trained on facilities D, F and G.

Figure D.13

Appendix E

Dataset statistics

Jupyter Notebooks which perform dataset profiling can be found on GitHub for dataset D[1], F[2] and G[3], respectively.

[1] https://github.com/hermanwh/master-thesis/blob/master/0_profiling_D.ipynb
[2] https://github.com/hermanwh/master-thesis/blob/master/0_profiling_F.ipynb
[3] https://github.com/hermanwh/master-thesis/blob/master/0_profiling_G.ipynb

Table E.1: Statistics, dataset D

Column	Mean	SD	Median	CV	Kurtosis	Skewness
20FT001	54.906	9.400	58.916	0.171	−0.552	−0.682
20PT001	40.000	0.000	40.000	NA	NA	NA
20PDT001	0.323	0.091	0.355	0.281	−0.769	−0.472
20TT001	87.973	4.833	89.400	0.055	−0.610	−0.398
20TT002	25.007	0.131	25.000	0.005	215.675	9.989
50FT001	277.033	122.273	261.030	0.441	−0.222	0.597
50PT001	5.000	0.000	5.000	NA	NA	NA
50PDT001	0.311	0.239	0.245	0.767	1.184	1.338
50TT001	10.543	0.749	10.400	0.071	−1.360	0.305
50TT002	20.164	2.452	19.700	0.122	−0.201	0.679
50TV001	0.462	0.204	0.435	0.441	−0.222	0.599

Table E.2: Statistics, dataset F

Column	Mean	SD	Median	CV	Kurtosis	Skewness
FYN0111	7.622	1.564	8.549	0.205	−1.216	−0.6016
TT0106MAY	59.036	4.432	58.796	0.0751	−0.698	0.024
TIC0105CAYX	24.036	0.292	24.007	0.0121	17.089	0.404
TIC0425CAYX	19.750	1.059	19.986	0.0536	8.0248	−2.217
TT0653MAY	54.265	3.420	53.587	0.063	1.304	−0.0293
TIC0105CAY	34.504	4.662	36.071	0.135	11.759	0.410

Table E.3: Statistics, dataset G

Column	Mean	SD	Median	CV	Kurtosis	Skewness
FI0010	3895.134	147.371	3860.079	0.038	11.261	2.609
PDI0064	1.173	0.179	1.154	0.153	−0.354	0.260
TZI0012	126.780	3.742	126.537	0.0295	0.677	0.555
TI0066	29.996	0.564	29.905	0.0188	50.821	4.847
FI0027	201.608	32.852	196.708	0.163	0.263	0.496
PI0001	10.051	0.231	10.001	0.023	21.355	4.491
PDT0024	3.505	1.438	3.387	0.410	−1.491	−0.047
TT0025	9.399	1.161	9.186	0.123	−0.765	0.235
TT0026	29.075	2.227	29.120	0.077	−0.360	0.032
TIC0022U	61.067	15.207	56.236	0.249	1.455	1.528