Universiteit
Leiden
The Netherlands

Unsupervised representation learning for time series anomaly detection and beyond

Ferreira Correia; L.

Citation

Unsupervised representation learning for time series anomaly detection and beyond. (2026, September 8). *Unsupervised representation learning for time series anomaly detection and beyond*. Retrieved from https://hdl.handle.net/1887/4309435

Version:	Publisher's Version
License:	Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden
Downloaded from:	https://hdl.handle.net/1887/4309435

Note: To cite this publication please use the final published version (if applicable).

Unsupervised Representation Learning for Time Series Anomaly Detection and Beyond

Proefschrift

ter verkrijging van
de graad van doctor aan de Universiteit Leiden,
op gezag van rector magnificus prof.dr. S. de Rijcke,
volgens besluit van het college voor promoties
te verdedigen op dinsdag 8 september 2026
klokke 14.30 uur

door

Lucas Ferreira Correia

geboren te Braga, Portugal

in 1998

Promotor:
Prof. dr. T.H.W. Bäck

Co-promotor:
Dr. A.V. Kononova

Promotiecomissie:
Prof. dr. M.M. Bonsangue
Prof. dr. A. Plaat
Dr. Y. Fan
Dr. N. van Stein
Prof. dr. K.M. Malan (University of South Africa)
Prof. dr. B. Sendhoff (Technical University Darmstadt)

Copyright © *2026 Lucas Ferreira Correia*

Cover: *Naila Lakehsar*, *The JetBrains Mono Project Authors*

Contents

Chapter 1

Introduction

1.1 Background

As the digitisation of industrial processes progresses, larger and larger datasets result from the data recorded. Ensuring this data is representative of the process is important, since downstream tasks like modelling or optimisation can be negatively impacted by incomplete or contaminated data. For tasks that require system behaviour modelling, data deviating from the norm is hence undesired, and we speak of *anomalous* behaviour. Recorded data manifests itself in many forms depending on the application and domain, one form being time series. Examples of real-world time series applications are diverse, ranging from the medical field [1] and server machines [2] to water distribution [3] and treatment [4]. Beyond that, the automotive industry also presents a real-world setting which can benefit from time series anomaly detection. Cars are complex dynamic systems whose behaviour can be described in many ways, one of which being the longitudinal dynamics, which is largely dictated by the powertrain within the vehicle. Automotive powertrains consist of multiple components, which contribute to a wide range of transient behaviour, including highly dynamic and more static signals.

1.2 Research Questions

This work attempts to answer a number of research questions relevant to the state of the art in the field of time series anomaly detection. Below, each question is posed and subsequently elaborated on, while being directly addressed in the following chapters.

Research Questions

RQ1 Can a non-trivial multivariate time series dataset be generated from physically motivated simulation models?

Measurable progress in the research field of unsupervised time series anomaly detection has been difficult to quantify over the last few years. While significant research volume has flowed into creating ever more complex data-driven models to approach the task, comparatively little work has been dedicated into benchmarking. Within benchmarking, publicly available data plays a large role, as it allows for the independent verification and reproduction of results. This contrasts with data stemming from real-world industrial processes, for instance, which often subject to strict levels of confidentiality and hence cannot not be published. Unfortunately, however, publicly available datasets have shown to be ill-suited due to several flaws and its distance from real-world problems in terms of dataset size and diversity. To answer this research question, the viability of physically inspired models for dataset generation is investigated in order to provide a publicly available dataset with real world-like complexity.

RQ2 Are satisfactory results obtainable when detecting anomalous behaviour in multivariate time series using completely unsupervised methods?

Although many studies in time series anomaly detection claim to propose unsupervised methods, only the model training is performed without labels, while adjacent processes like threshold or window size choice are informed with some amount of labelled data. In the real world, labelled data is either not available or expensive to obtain, and hence truly unsupervised methods are required for deployment. Clearly, proposed methods that use labelled data do not qualify as truly unsupervised and cannot as easily be deployed in the real world. To answer this research question, a novel, completely unsupervised method is benchmarked against the state of the art from literature to better understand its effectiveness in the real world.

RQ3 How well can a time series anomaly detection approach generalise to unseen data stemming from the same dynamic system?

In the real world, training data may not capture all possible scenarios due to very recent start of data collection, for example. Additionally, it may even not be possible to cover all scenarios due to random external effects, say for instance, caused by driving a different route to a destination every time with different levels of traffic and weather conditions. As is evident, for time series anomaly detection

in such settings, approaches that can generalise based on limited training data are desirable, and hence this aspect is further investigated.

RQ4 Can active learning be harnessed to find a suitable threshold for completely unsupervised methods?

Contamination of the training subset with anomalous data can pose major challenges for truly unsupervised time series anomaly detection approaches. Not only does the modelling process become more difficult, but so does the threshold choice. Truly unsupervised approaches are observed to use a far-from-ideal threshold due to the presence of anomalous data and hence the use of active learning, i.e. intelligently querying a domain expert to label data, is investigated to improve the threshold choice.

RQ5 How robust are active learning-based thresholds to mislabelling by the oracle?

In the real world, domain experts acting as oracles may still make mistakes when labelling time series data. Therefore, different mislabelling rates are experimented with to simulate such cases in an effort to study the impact on the anomaly detection performance.

1.3 Structure of this Thesis and Publications

This thesis is structured as follows. First, Chapter 2 provides the reader with relevant background in the field of time series anomaly detection, as well as with the theory behind key concepts discussed in this work. Following that, Chapter 3 is largely based on a survey [5] of the research area. It not only introduces methods proposed for time series anomaly detection but also provides a perspective on the state of benchmarking in the field. Chapter 4 addresses RQ1 by presenting a publicly available dataset [6] aiming to replicate real-world complexity with its physically motivated nature. Furthermore, a real-world dataset is also introduced along with some preliminary analysis revealing key properties within both datasets. Chapter 5 then provides the reader with a range of experiments and the associated results representing the viability of online and unsupervised detection methods and their capability to generalise, therefore addressing RQ2 and RQ3, respectively. This chapter is based on two publications. One paper [7] proposes a novel unsupervised data-driven model, tested on an offline real-world use-case, and another [8] compliments the findings by extending the problem to an online scenario and proposing a novel set of metrics for evaluation. Then, Chapter 6 is mostly based on a publication [9] which further addresses some of the issues

identified in Chapter 5 revolving around threshold choice and attempts to provide an answer to RQ4 and RQ5 by exploring the combination of active learning with unsupervised anomaly detection methods. Furthermore, another survey [10] under review identifies a research gap within predictive maintenance, which is briefly touched upon in Chapter 7. It illustrates how modelling dynamic system behaviour in a data-driven way can be applied beyond anomaly detection to fields like predictive maintenance in the case of a real-world problem. Finally, Chapter 8 summarises the main body of this thesis and concisely answers the research questions posed above, while also outlining potential future work. The full citations for the publications included in this thesis are provided below.

[5] Lucas Correia, Jan-Christoph Goos, Philipp Klein, Thomas Bäck, and Anna V. Kononova. "Online model-based anomaly detection in multivariate time series: Taxonomy, survey, research challenges and future directions". In: *Engineering Applications of Artificial Intelligence* 138 (2024), p. 109323. DOI: 10.1016/j.engappai.2024.109323

[6] Lucas Correia, Jan-Christoph Goos, Thomas Bäck, and Anna V. Kononova. "PATH: A Discrete-sequence Dataset for Evaluating Online Unsupervised Anomaly Detection Approaches for Multivariate Time Series". In: *Big Data Research* (2025), p. 100573. DOI: 10.1016/j.bdr.2025.100573

[7] Lucas Correia, Jan-Christoph Goos, Philipp Klein, Thomas Bäck, and Anna Kononova. "MA-VAE: Multi-Head Attention-Based Variational Autoencoder Approach for Anomaly Detection in Multivariate Time-Series Applied to Automotive Endurance Powertrain Testing". In: *Conference on Neural Computation Theory and Applications*. SciTePress, 2023, pp. 407–418. DOI: 10.5220/0012163100003595

[8] Lucas Correia, Jan-Christoph Goos, Philipp Klein, Thomas Bäck, and Anna V. Kononova. "TeVAE: A Variational Autoencoder Approach for Discrete Online Anomaly Detection in Variable-State Multivariate Time-Series Data". In: *Computational Intelligence*. Vol. 1196. Series Title: Studies in Computational Intelligence. Springer Nature Switzerland, 2025, pp. 106–132. DOI: 10.1007/978-3-031-85252-7_7

[9] Lucas Correia, Jan-Christoph Goos, Thomas Bäck, and Anna V. Kononova. "DQS: A Low-Budget Query Strategy for Enhancing Unsupervised Data-driven

Anomaly Detection Approaches". In: *Journal of Big Data* (2026). DOI: 10.1186/s40537-026-01524-3.

[10] Bernd G Wagner, Qi Huang, Lucas Correia, Thomas Bäck, Anna Kononova, and Niki Van Stein. *REMAINING USEFUL LIFE IN COMPLEX MULTI-COMPONENT SYSTEMS: TAXONOMY, REVIEW, AND RESEARCH DIRECTIONS*. 2025. DOI: 10.36227/techrxiv.176288069.90065228/v1.

Chapter 2

Background and Theory

To aid the reader in understanding all concepts discussed in this thesis, this chapter provides an introduction to the principles required. First, Chapter 2.1 offers a brief overview on time series theory, highlighting some key properties of time series data and how to interpret them. Then, Chapter 2.2 presents the required foundations on anomaly detection as well as a novel taxonomy. Chapters 2.3–2.5 introduce a number of time series modelling constructs which play a foundational role later on in Chapter 5, while the concepts presented in Chapters 2.6 and 2.7 become especially relevant in Chapter 6.

2.1 Time Series Theory

Time series are signals that represent a property or feature of a dynamic system as a function of time, usually sampled at a fixed rate. Unlike image or text data, which humans learn to process from a young age, time series data is less intuitive and requires corresponding domain knowledge to be interpreted. An arbitrary time series $\mathcal{S}$ can be univariate, i.e. $\mathcal{S} \in \mathbb{R}^T$, or multivariate, i.e. $\mathcal{S} \in \mathbb{R}^{T \times d}$, where T refers to the number of discrete time steps and d to the number of features in the time series. More specifically, univariate time series can only possess a temporal correlation, i.e. along the time axis, whereas multivariate time series can also contain correlation along the feature axis.

Time series can be decomposed into different components. The first one is noise, also referred to residuals, in which observations are independent of one another, identically distributed and have a mean of zero [11]. Another component is trend, which is

present when a time series has a non-zero mean. If the mean of the time series is zero, it is referred to as a stationary time series [11, 12]. The last component is seasonality, which is present when certain patterns occur at arbitrary frequencies [11].

Time series can be analysed using different methods in the time domain and in the frequency domain.

One time domain method is the autocorrelation analysis. Calculating the autocorrelation can be a helpful tool to understand how time steps are correlated with one another at a given lag. To calculate the autocorrelation ρ_h at a given lag h, the autocovariance γ_h needs to be obtained first, which is shown in Equation 2.1 [12].

$$\gamma_h = \mathbb{E}\left[(\mathcal{S}_{t+h} - \bar{\mathcal{S}})(\mathcal{S}_t - \bar{\mathcal{S}})\right] \tag{2.1}$$

Here, $\mathcal{S}$ denotes an arbitrary univariate time series such that $\mathcal{S} \in \mathbb{R}^T$, and $\bar{\mathcal{S}}$ denotes the mean of the time series $\mathcal{S}$. As is evident, for $h = 0$, the autocovariance is simply the variance. With the autocovariance established, the autocorrelation is defined as the autocovariance normalised by the variance, as shown in Equation 2.2 [12].

$$\rho_h = \frac{\gamma_h}{\gamma_0} \tag{2.2}$$

Time series which feature a high autocorrelation for a large number of lags tend to feature slower dynamics than time series featuring high autocorrelation for a more limited number of lags.

In contrast, a method in the frequency domain is a Fourier analysis. The Fourier representation of a signal decomposes it into its different sinusoidal components, revealing the different frequencies present in the time series [12]. For digital and therefore discretely sampled signals, a discrete Fourier transform F needs to be applied, which is mathematically described in Equation 2.3 [12].

$$F_k = \frac{1}{\sqrt{T}} \sum_{t=1}^{T} \mathcal{S}_t e^{-i2\pi t \frac{k}{T}} \tag{2.3}$$

Here, F_k denotes the presence of the frequency k in time series $\mathcal{S}$.

2.2 Anomaly Detection

To understand anomaly detection problems, one must become familiar with their multi-faceted nature. To provide the reader with an illustration of the relationship be-

tween terms introduced in the taxonomy used, an overview is represented graphically in Figure 2.1.

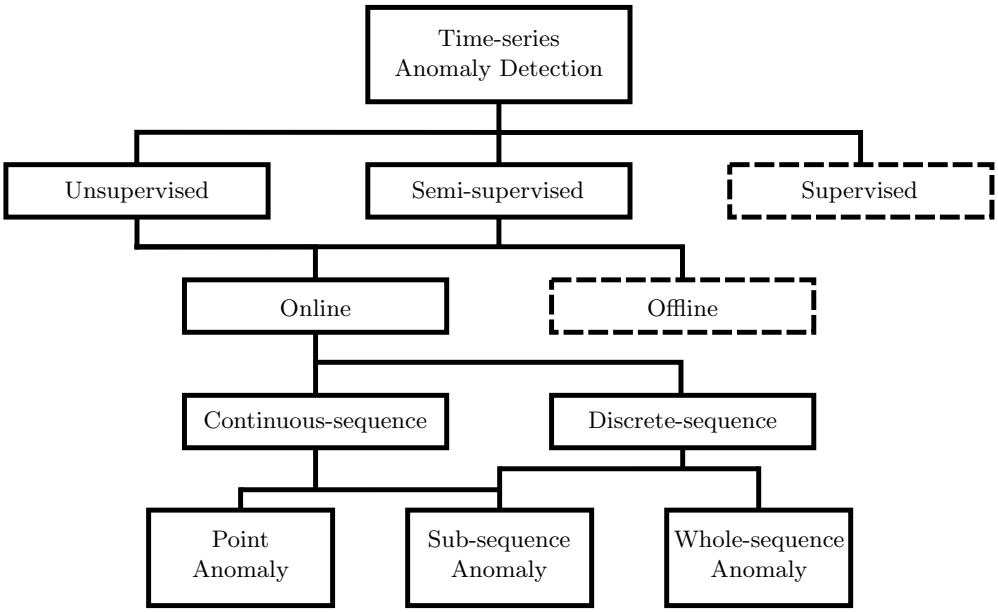

Figure 2.1: A visual representation of taxonomy used in this thesis.

2.2.1 Anomaly Types

Anomalies in time series can be assumed to have different shapes and forms. A commonly used and accepted definition [13] reads as follows:

"An observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism."

The *mechanism* can be a change in the system, like a broken component inside the system under investigation. Over the years, several taxonomies are proposed to better classify different anomalies. The taxonomy here follows the one suggested by Blásquez-García et al. [14], where anomalies are classified into point outliers, sub-sequence outliers (also known as collective anomalies in the literature) and outlier time series. While they acknowledge that the terms *outlier* and *anomaly* are widely used to refer to the same concept, Blásquez-García et al. [14] note that outliers are mostly used in cases of unwanted patterns whereas anomalies in cases of interest. In this work, these

three types will be from now on referred to as *point anomalies*, *sub-sequence anomalies* and *whole-sequence anomalies*, respectively.

Consider a testing subset $\mathcal{D}^{\text{test}}$ in a data set $\mathcal{D}$ containing N sequences, such that $\mathcal{D}^{\text{test}} = \{\mathcal{S}_1, ..., \mathcal{S}_n, ..., \mathcal{S}_N\}$, where $\mathcal{S}_n \in \mathbb{R}^{T_n \times d_{\mathcal{D}}}$. In a given multivariate sequence $\mathcal{S}_n$, also consider an anomaly event $\mathcal{A} \in \mathbb{R}^{S \times d_{\mathcal{A}}}$ of length S that is detected in $d_{\mathcal{A}}$ channels, where $d_{\mathcal{A}} \leq d_{\mathcal{D}}$.

A *point anomaly* is defined as a value at a time step that does not conform to the typical behaviour of a system, or, in other words, $S = 1$ for $\mathcal{A}$. An example of a point anomaly is illustrated in Figure 2.2. While more easily detectable than the other

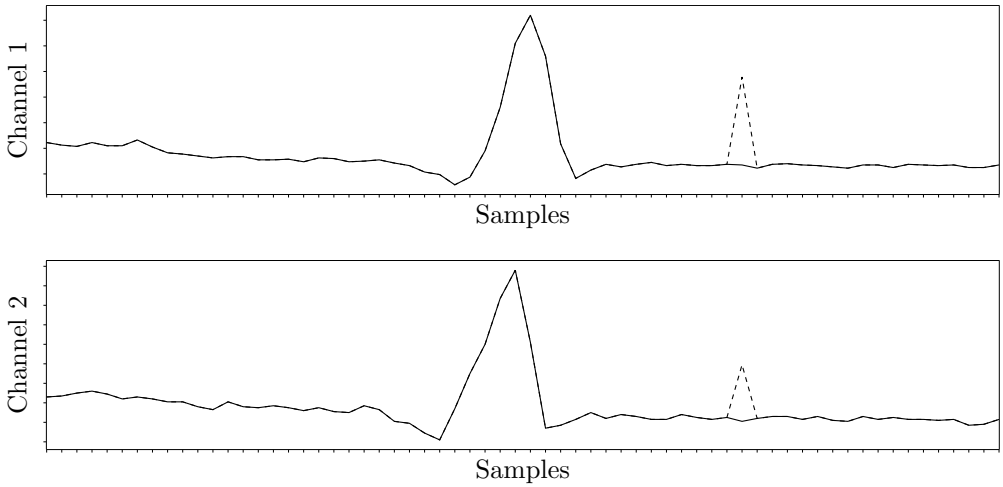

Figure 2.2: Example of a *point anomaly* in both channels. The continuous line represents the ground-truth time series and the dashed line the anomaly. The ground-truth time series shown is a heartbeat from the MIT-BIH Arrhythmia Database [1].

types, these anomalies are rare events in the real world, as systems affected cannot usually return to a nominal state before the next time step arrives unless the sampling rate is very low. Throughout this work, *nominal* is used as a synonym for *normal* or *anomaly-free* to avoid confusion with normal/Gaussian distributions.

Sub-sequence anomalies are defined by a series of anomalous time steps, i.e. a sub-sequence within a time series that does not reflect the nominal behaviour of a system, or, in other words, $1 < S < T_n$ for $\mathcal{A}$. In a real-world system, this type of anomaly may occur when a component in the said system runs at reduced functionality or fails but manages to recover to a nominal state after a period of time. An example of a sub-sequence anomaly is illustrated in Figure 2.3.

A *whole-sequence anomaly* can be seen as a long sub-sequence that has the same

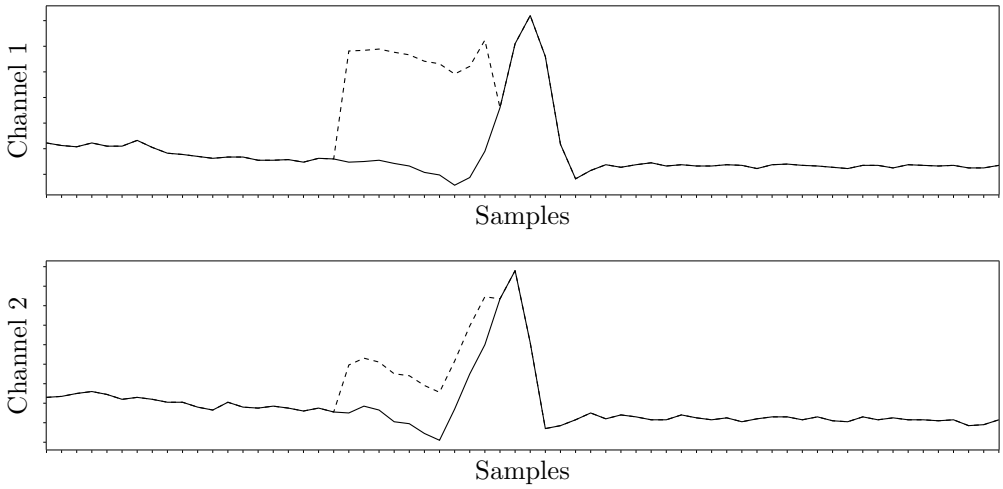

Figure 2.3: Example of a *sub-sequence anomaly* in both channels. The continuous line represents the ground-truth time series and the dashed line the anomaly. The ground-truth time series shown is a heartbeat from the MIT-BIH Arrhythmia Database [1].

length as the entire sequence, or, in other words, $S = T_n$ for $\mathcal{A}$. This type of anomaly can occur when an initial parameter or state deviates from the norm, leading to all observations in the sequence being anomalous as well. An example of a whole-sequence anomaly is illustrated in Figure 2.4.

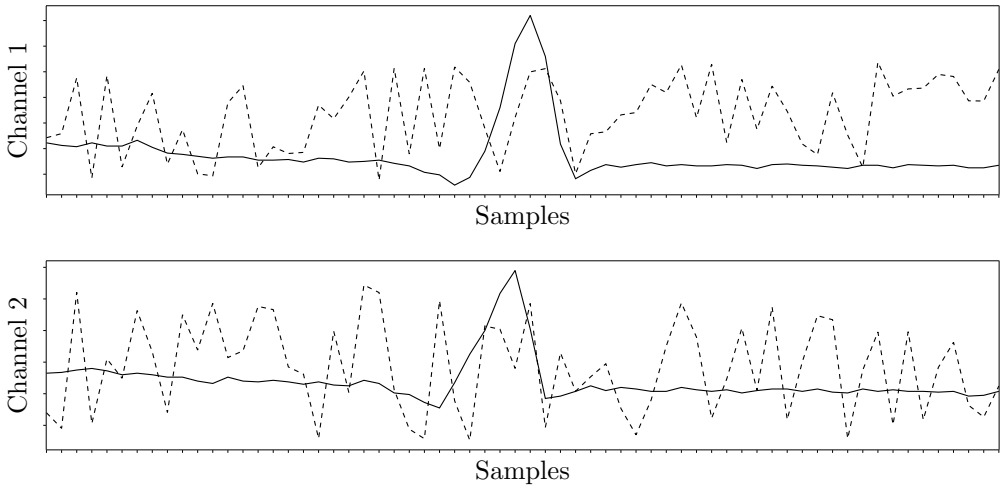

Figure 2.4: Example of a *whole-sequence anomaly* in both channels. The continuous line represents the ground-truth time series and the dashed line the anomaly. The ground-truth time series shown is a heartbeat from the MIT-BIH Arrhythmia Database [1].

2.2.2 Concept of Supervision

Analogous to types of learning, there is supervised, semi-supervised and unsupervised anomaly detection. *Supervised* anomaly detection is essentially imbalanced binary time series classification and is only rarely found in literature. This is most likely because, in real-world use cases, possible anomaly types are rarely known a priori. In addition to that, labelling data is expensive, which is why unsupervised and semi-supervised anomaly detection are more relevant in both literature and the real world. *Unsupervised* anomaly detection is independent of any labels, i.e. any available data for model training may contain both anomalous and nominal time series, and it is not known which is which [15]. In contrast to that, *semi-supervised* anomaly detection can be considered a more relaxed setting, where anomalous time series are absent from the training subset [15]. In the real world, this still requires some labelling to ensure an entirely nominal training subset, which is not always given. Some literature diverges from this taxonomy, agreeing with the supervised and semi-supervised definitions but defining unsupervised anomaly detection differently. According to Schmidl et al. [16], unsupervised anomaly detection involves detecting anomalous behaviour without a training procedure. This definition implies that learning nominal behaviour from *mostly* nominal data is not possible, which is investigated in Chapter 5.3.

2.2.3 Continuous- and Discrete-sequence Problems

Detecting anomalous behaviour in time series is referred to as *time series anomaly detection*, which can be split into two main areas: continuous- and discrete-sequence, where the former is the most common type present in public datasets. It is defined as detecting anomalies in a process that runs for a continuous time period without breaks. Typically, the test subset $\mathcal{D}^{\text{test}}$ in the dataset $\mathcal{D}$ in a continuous-sequence problem consists of a singular multivariate time series composed of multiple nominal and anomalous sub-sequences, i.e. $\mathcal{D}^{\text{test}} = \{\mathcal{S}_1\}$. Discrete-sequence anomaly detection, in contrast, is defined as detecting anomalies in N chunks of processes that happen independently of each other, such as automotive test benches, where several tests may occur sequentially but are not temporally contiguous and hence provide a time series for each test, i.e. $\mathcal{D}^{\text{test}} = \{\mathcal{S}_1, ..., \mathcal{S}_n, ..., \mathcal{S}_N\}$. Here, the testees, i.e. the test subjects, are not monitored over a continuous period of time, but are instead monitored solely during each process chunk. Automotive testing is not the only use for discrete-sequence anomaly detection, however. Another discrete-sequence problem could also include the analysis of the flight behaviour of an aeroplane, where the time while it is

docked is irrelevant and may not be recorded. Therefore, datasets for discrete-sequence anomaly detection consist of several nominal and anomalous time series, where a given anomalous time series may be entirely anomalous, in case of a whole-sequence anomaly, or only partly, in case of a point or sub-sequence anomaly.

2.2.4 Online Anomaly Detection

Detecting anomalous behaviour in a timely manner is also advantageous, since the source of anomalous behaviour may bring about damage to said system. Such problems require a time series to be evaluated as it is being recorded, not just once it is finished; such problems are referred to as *online* time series anomaly detection. Using model-based approaches generally involves two processes: training and inference. Training is the process of automatically adjusting model parameters by means of optimisation using training data, whereas inference refers to the application, where model parameters are no longer adjusted. Online approaches are defined as models that perform inference in an online manner and, optionally, are trained in an online manner. Online training, therefore, refers to the process of adjusting model parameters as training data is streamed. Unlike offline training, which is done once in most cases, online training runs continuously and for an undefined amount of time. Online inference refers to the ability of an approach to detect anomalous behaviour in time steps as soon as they are observed, rather than waiting for the time series to have finished streaming and only then evaluating the entire sequence, i.e. offline. This is especially useful in real-world use cases where timely detection is important. A more strict version of online anomaly detection is real-time anomaly detection, where the current time step is evaluated before the next one arrives.

2.3 Long Short-term Memory Cells

Introduced in 1997 by Hochreiter and Schmidhuber [17], long short-term memory (LSTM) cells were developed to prevent gradients from exploding or disappearing when propagating backwards in time during training, a problem associated with regular recurrent reural retworks (RNN) cells. This architecture still suffered from a problem, where cell states may increase uncontrollably and hence an evolution featuring so-called forget gates was introduced by Gers et al. [18] to avoid saturation of the output activation function. The LSTM cell consists of four gates, which compute the current cell state c_t and the hidden state h_t at time step t. The hidden state is used as the

output of the cell to other downstream layers, whereas the cell state is used as the memory of the cell, which includes relevant information from previous time steps and enables LSTM cells to represent temporal dependencies. Other architectures, like the gated recurrent unit (GRU) [19] combine the hidden and cell states into one state, due to their overlapping functionality. Consider an arbitrary multivariate time series $\mathcal{S} \in \mathbb{R}^{T \times d_{\mathcal{D}}}$ and an LSTM layer it is fed into. The hidden and cell states are computed by feeding the input at the current time step $\mathcal{S}_t$ into the forget gate, the cell candidate gate, the input gate and the output gate. The inner connections between the different gates inside an LSTM cell is graphically depicted in Figure 2.5. These gates are

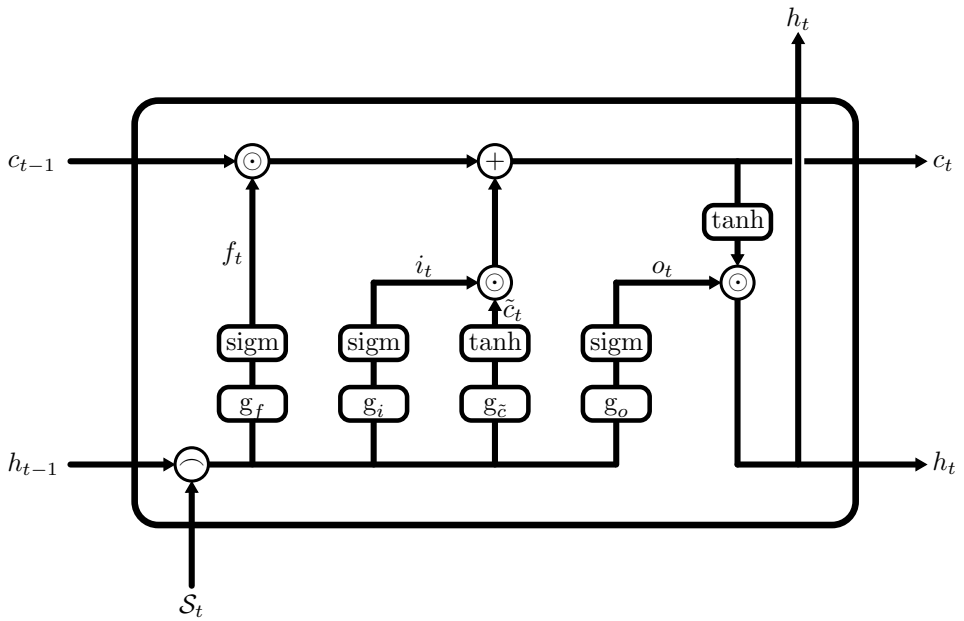

Figure 2.5: Illustration of the topology of a long short-term memory cell, with $\mathcal{S}_t$, c_t and h_t denoting the current input time step, cell state and hidden state at time step t, respectively. Moreover, $\odot$ denotes the Hadamard product and $\frown$ denotes the concatenation of two vectors.

parametrised by three sets: input weights W, the recurrent weights R and the input bias b, as shown in Equation 2.4.

$$W = \begin{bmatrix} W_i \\ W_f \\ W_{\tilde{c}} \\ W_o \end{bmatrix} \qquad R = \begin{bmatrix} R_i \\ R_f \\ R_{\tilde{c}} \\ R_o \end{bmatrix} \qquad b = \begin{bmatrix} b_i \\ b_f \\ b_{\tilde{c}} \\ b_o \end{bmatrix} \tag{2.4}$$

The forget gate controls the level of reset of the previous cell state c_{t-1}. It outputs forget gate mask f_t denoting how open the gate is to passing information and is modelled using Equation 2.5.

$$f_t = \sigma(\mathrm{g}_f) = \sigma(W_f \cdot \mathcal{S}_t + R_f \cdot h_{t-1} + b_f) \tag{2.5}$$

The resulting forget gate output f_t then multiplied by the cell state from the previous time step c_{t-1} to mask information deemed unimportant.

The cell state candidate $\tilde{c}_t$ contains new potential information that will be added and is computed as shown in Equation 2.6.

$$\tilde{c}_t = \tanh(\mathrm{g}_{\tilde{c}}) = \tanh(W_{\tilde{c}} \cdot \mathcal{S}_t + R_{\tilde{c}} \cdot h_{t-1} + b_{\tilde{c}}) \tag{2.6}$$

The input gate controls the level of update of the cell state candidate $\tilde{c}_t$ and protects the memory contents from perturbation by irrelevant inputs and forward-flowing activation [17, 18]. Its output is the input gate mask i_t which is calculated using Equation 2.7.

$$i_t = \sigma(\mathrm{g}_i) = \sigma(W_i \cdot \mathcal{S}_t + R_i \cdot h_{t-1} + b_i) \tag{2.7}$$

The product of the cell candidate $\tilde{c}_t$ and the input gate mask i_t is then added to the masked previous cell state and becomes the current cell state c_t. Hence, the current cell state c_t can be expressed using Equation 2.8.

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{2.8}$$

Here, $\odot$ designates the Hadamard Product.

The output gate fulfils a similar role to the forget gate. The output of this gate is used to protect the cell from backward flowing error [18]. This can be represented by Equation 2.9.

$$o_t = \sigma(\mathrm{g}_o) = \sigma(W_o \cdot \mathcal{S}_t + R_o \cdot h_{t-1} + b_o) \tag{2.9}$$

The current cell state c_t is then normalised by a hyperbolic tangent function and then element-wise multiplied with the output gate mask o_t to retain the information needed. The product is the current hidden state h_t, Equation 2.10.

$$h_t = o_t \odot \tanh(c_t) \tag{2.10}$$

Evidently, if the LSTM layer discussed above is placed after another LSTM layer, its input is no longer the current time step $\mathcal{S}_t$ of the input time series, but rather the current hidden state h_t of the previous layer.

An LSTM layer in its "unrolled" state can be thought of a chain of such LSTM cells, all identically parameterised, which pass the computed cell- and hidden states to one another to model temporal dependencies. Another way to think about an LSTM layer is in its "rolled" state as a single LSTM cell, where its outputs are fed back as its inputs.

Bidirectional long short-term memory (BiLSTM) layers are the composition of two parallel LSTM layers, each modelling time in a different direction, whose hidden states are concatenated before being passed to the following layer.

2.4 Variational Autoencoders

The variational autoencoder (VAE) [20, 21] is a generative model that structurally resembles an autoencoder, but is theoretically derived from variational Bayesian statistics. In contrast to the regular deterministic autoencoder, the VAE uses the evidence lower bound (ELBO), which is a lower bound approximation of the so-called log evidence $\log p_\theta(\mathbf{X})$, as its objective function. The ELBO, Equation 2.11, can be expressed as the reconstruction log-likelihood and the negative Kullback-Leibler divergence D_{KL} between the approximate posterior $q_\phi(\mathbf{Z}|\mathbf{X})$ and the prior $p_\theta(\mathbf{Z})$, which is typically assumed to be a Gaussian distribution [22]. The training data does not have to follow a Gaussian distribution, since the latent distribution is a nonlinear mapping of the training data.

$$\mathcal{L}_{\theta,\phi}(\mathbf{X}) = \mathbb{E}_{\mathbf{Z}\sim q_\phi(\mathbf{Z}|\mathbf{X})}\left[\log p_\theta(\mathbf{X}|\mathbf{Z})\right] - D_{\mathrm{KL}}(q_\phi(\mathbf{Z}|\mathbf{X})||p_\theta(\mathbf{Z})) \tag{2.11}$$

Here, $\mathbf{Z} \in \mathbb{R}^{w\times d_\mathbf{Z}}$ is the sampled latent matrix, $\mathbf{X} \in \mathbb{R}^{w\times d_\mathcal{D}}$ denotes the input window and w represents the window length, whereas $d_\mathcal{D}$ and $d_\mathbf{Z}$ refer to the dataset and latent matrix dimensionality, respectively. Gradient-based optimisation minimises an objective function and the goal is the maximisation of the ELBO, and hence the final loss function is defined as the negative of Equation 2.11, shown in Equation 2.12.

$$\mathcal{L}_{\mathrm{VAE}} = -\mathcal{L}_{\theta,\phi}(\mathbf{X}) \tag{2.12}$$

Finally, to enable the backpropagation through the otherwise intractable gradient

of the ELBO, the *reparameterisation trick* [20] is applied, shown in Equation 2.13. This is one of the reasons for the use of a Gaussian distribution as the latent distribution, although any distribution type from the "location-scale" type can be used [20], like a Laplace distribution.

$$\mathbf{Z} = \boldsymbol{\mu}_{\mathbf{Z}} + \epsilon \cdot \boldsymbol{\sigma}_{\mathbf{Z}} \tag{2.13}$$

Note that ϵ is simply sampled from a standard Gaussian distribution, such that $\epsilon \sim \mathcal{N}(0, 1)$ and $(\boldsymbol{\mu}_{\mathbf{Z}}, \log \boldsymbol{\sigma}^2_{\mathbf{Z}}) = q_\phi(\mathbf{X})$.

2.5 Multi-head Attention

To simplify the explanation of multi-head attention (MA) as employed in this work, multi-head self-attention (MS) will be described instead, with the small difference between MA and MS being pointed out at the end. MS consists of two different concepts: self-attention and its multi-head extension. Self-attention is nothing more than scaled dot-product attention [23] where the key, query, and value are the same. The scaled dot-product attention score is the softmax [24] of the product between query matrix $\mathbf{Q}$ and key matrix $\mathbf{K}$, which is scaled by $\sqrt{d_{\mathbf{K}}}$. The product between the attention score and the value matrix $\mathbf{V}$ yields the context matrix $\mathbf{C}$, as shown in Equation 2.14.

$$\mathbf{C} = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_{\mathbf{K}}}}\right)\mathbf{V} \tag{2.14}$$

Compared to recurrent or convolutional layers, self-attention offers a variety of benefits, such as the reduction of computational complexity, as well as an increased amount of operations that can be parallelised [23]. Moreover, self-attention inherits an advantage over Bahdanau-style attention [25] from the underlying scaled dot-product attention mechanism: it can run efficiently in matrix multiplication manner [23].

Multi-head self-attention then allows the attention model to attend to different representation subspaces [23], in addition to learning useful projections, rather than being a stateless transformation [26]. This is achieved using weight matrices $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$, $\mathbf{W}_i^V$, which contain trainable parameters and are unique for each head i, as shown in Equation 2.15.

$$\mathbf{Q}_i = \mathbf{Q}\mathbf{W}_i^Q \qquad \mathbf{K}_i = \mathbf{K}\mathbf{W}_i^K \qquad \mathbf{V}_i = \mathbf{V}\mathbf{W}_i^V \tag{2.15}$$

Once the query, key and value matrices are linearly transformed via the weight ma-

trices, the context matrix $\mathbf{C}_i$ for each head i is computed using Equation 2.16.

$$\mathbf{C}_i = \text{Softmax}\left(\frac{\mathbf{Q}_i\mathbf{K}_i^T}{\sqrt{d_\mathbf{K}}}\right)\mathbf{V}_i \tag{2.16}$$

Then, for h heads, the different context matrices are concatenated and linearly transformed again via the weight matrix $\mathbf{W}^O$, resulting in the multi-head context matrix $\mathbf{C} \in \mathbb{R}^{w \times d_\mathbf{z}}$, Equation 2.17.

$$\mathbf{C} = [\mathbf{C}_1, ..., \mathbf{C}_h]\mathbf{W}^O \tag{2.17}$$

The underlying mechanism of MA is identical to MS, with the only difference being that $\mathbf{K} = \mathbf{Q} \neq \mathbf{V}$. Essentially, MA finds which time steps correlate most with each other inside a given input window and weighs the time steps in context matrix $\mathbf{C}$ accordingly. The benefit of this alteration is investigated in Chapter 5.3.

2.6 Active Learning

The oldest literature in the field of active learning deals mostly with data augmentation for supervised classification algorithms, though lately, it has also found application in anomaly detection. In the context of model-based anomaly detection, active learning typically involves two components in a loop, as shown in Figure 2.6.

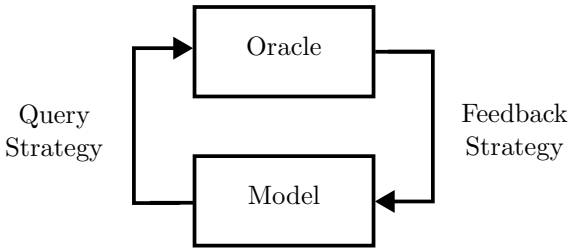

Figure 2.6: Interaction between the oracle and a model-based anomaly detector.

The first component is the oracle, also referred to as a user, human annotator, or domain expert, whose task is to label previously unlabelled samples as nominal or anomalous, so they can be used for other tasks. The second component is the anomaly detection model, which is trained in an unsupervised manner, i.e. on unlabelled data, or in a semi-supervised manner, i.e. on nominal-only data. The samples labelled by the oracle are used to calibrate the model to improve detection performance. The oracle and the model interact with each other using the query strategy (QS) and the

feedback strategy (FS).

The QS is defined as the procedure used to choose the samples to be sent to the oracle [27]. The goal of the QS is to select the samples from a candidate pool that provides the most useful information for the FS. Labelling is expensive, and hence the QS is often constrained by a parameter called the query budget, which represents the number of samples an oracle is willing to label at a time.

The FS is defined as the method of applying the knowledge in form of the labelled samples. In the case of data augmentation supervised classification algorithms, the FS is straight-forward, however, in the unsupervised anomaly detection, it plays a larger role due to the inherent incompatibility of unsupervised methods with labelled data.

2.7 Dynamic Time Warping

Originally proposed by Vintsyuk [28], Sakoe and Chiba [29], dynamic time warping (DTW) is used to calculate the optimal alignment of two time series that may be of different lengths and out of sync w.r.t. each other.

To calculate the DTW distance δ, consider two univariate sequences $\mathbf{x}$ and $\mathbf{y}$, where $\mathbf{x} \in \mathbb{R}^{T_\mathbf{x}}$, $\mathbf{y} \in \mathbb{R}^{T_\mathbf{y}}$, and u, v represent arbitrary indices within the respective sequences, such that $\mathbf{x} = [x_1, ..., x_u, ..., x_{T_\mathbf{x}}]$ and $\mathbf{y} = [y_1, ..., y_v, ..., y_{T_\mathbf{y}}]$. The local cost matrix $\mathbf{G}$ represents the cost of aligning each time step in $\mathbf{x}$ with each time step in $\mathbf{y}$. The implementation of DTW used in this work [30] uses the Euclidean norm for the computation of the local cost. Hence, the local cost between $\mathbf{x}$ at index u and $\mathbf{y}$ at index v is as shown in Equation 2.18.

$$\mathbf{G}_{u,v} = (x_u - y_v)^2 \tag{2.18}$$

The optimal path through $\mathbf{G}$ represents the path where the cumulative cost is lowest and can be found using dynamic programming. In the cumulative cost matrix $\mathbf{D}$, each index denotes the cumulative cost of arriving at the respective index; it is calculated as shown in Equation 2.19.

$$\mathbf{D}_{u,v} = \mathbf{G}_{u,v} + \min \begin{cases} \mathbf{D}_{u-1,1} & \text{for } u > 1 \\ \mathbf{D}_{1,v-1} & \text{for } v > 1 \\ \mathbf{D}_{u-1,v-1} & \text{for } u, v > 1 \end{cases} \tag{2.19}$$

Finally, the DTW distance δ equals the square root of the cumulative cost of the

optimal path, as shown in Equation 2.20.

$$\delta = \sqrt{\mathbf{D}_{T_{\mathbf{x}},T_{\mathbf{y}}}} \tag{2.20}$$

Chapter 3

Related Work

In this chapter, the state-of-the-art related work in the field of time series anomaly detection is provided. First, the aspect of benchmarking published research is touched upon in Chapter 3.1, which encompasses an overview of the commonly used datasets in the literature, as well as an introduction of evaluation metrics used to quantify anomaly detection performance. Then, a condensed review of proposed approaches in the research field is provided in Chapter 3.2, along with an analysis of the existing research gaps.

3.1 Benchmarking Time Series Anomaly Detection Methods

Benchmarking is an important part of anomaly detection research, as it enables objectively measuring progress in the field. To benchmark any given approach against another, a standardised procedure needs to be agreed upon, which ideally isolates the approaches in question as the only variables during an experiment. For this to be achieved, the same version of a given dataset and the same metrics to evaluate detection performance need to be used. Another way to further isolate approaches as the only variable is to separate the threshold setting from the approach itself by quantifying detection performance solely using uncalibrated evaluation metrics [16], which are further explored in this chapter. This section provides an overview of datasets and evaluation metrics most used in the literature, and identifies research gaps that still need to be addressed.

3.1.1 Datasets

As discussed in Chapter 2.1, time series anomaly detection can be split into continuous-sequence problems and discrete-sequence problems, depending on the application and therefore structure of the data. The majority of publicly available datasets represent continuous-sequence problems and hence most approaches and evaluation metrics proposed in the literature focus solely on such problems, though as argued in Chapter 2.1, there are many industry-relevant applications, which would benefit from more research contributed to the discrete-sequence area.

The continuous-sequence dataset landscape can be largely segmented into two distinct categories: pre- and post-Wu and Keogh [31]. Wu and Keogh [31] argue that the most popular datasets are unsuitable to benchmark approaches against one another, highlighting four primary flaws, with most datasets exhibiting at least one of these issues:

- triviality
- unrealistic anomaly density
- uncertain labels
- run-to-failure bias

They define *triviality* as the ability of a problem to be solved with minimal code, thereby undermining the justification for complex, parameter-heavy models. They also highlight the issue of *unrealistic anomaly density*, where the common assumption that anomalies are rare events does not hold true in most cases. Another concern raised is the presence of *mislabelled ground truth*. In some instances, regions exhibiting similar behaviour are inconsistently labelled, sometimes as anomalous, other times as nominal, which can distort evaluation results. Finally, they identify a *run-to-failure bias* in certain datasets, where anomalies predominantly occur at the end of sequences. This pattern can artificially inflate anomaly detection performance due to models being biased towards predicting the later time steps as positive. It could be argued, however, that run-to-failure bias is an inherent characteristic of some real-world dynamic systems, especially those, where anomalous behaviour is mainly caused due to a component that breaks and therefore transitions from nominal operation to an anomalous state. In other domains, like cybersecurity, anomalous behaviour can be caused due to malicious attacks, which, when over, can lead to nominal behaviour after an anomaly. In discrete-sequence problems, run-to-failure bias is also less of an issue, due to the majority of sequences being nominal.

Before the work by Wu and Keogh [31], the majority of time series anomaly detection approaches were evaluated on a subset of five datasets: the secure water treatment (SWaT) dataset [4], the water distribution (WADI) dataset [3], the soil moisture active passive (SMAP) and Mars Science Laboratory (MSL) datasets [32], and the server machine dataset (SMD) [2]. They are listed in tabular form along with their key properties in Table 3.1.

Table 3.1: Key properties of the most popular datasets, where $d_{\mathcal{D}}$ refers to the number of features of the time series signals in the dataset, $\sum |\mathcal{S}_n|$ to the total number of time steps in the test subset and $\%A$ to the number of anomalous time steps in relation to all test time steps.

Name	$d_{\mathcal{D}}$	$\sum_{}^{N} \lvert\mathcal{S}_n\rvert$	$\%A$
SWaT [4]	51	449,919	12 %
WADI [3]	123	17,287	6 %
SMAP [32]	25	427,617	13 %
MSL [32]	55	73,729	11 %
SMD [2]	38	708,420	4 %

Wu and Keogh [31] only analyse SMAP, MSL and SMD datasets in their work, however, their criticism can be extended to the SWaT and WADI datasets too, while both suffer from a different issue as well.

The SWaT [4] and WADI [3] datasets are both published by researchers at the Singapore University of Technology and Design. Both consist of data from a water processing test bed under nominal conditions and under various attack scenarios, though no training/testing subset splits are provided. One of the problems with SWaT is that it contains six features that are constant throughout the entire dataset, making them redundant for modelling. In addition to that, the dataset features a very high anomaly density, in fact, the single anomalous subsequence between time steps 227828 and 262727 accounts for around 65 % of the total anomalous time steps in the dataset [5, 33]. This anomaly falls under the triviality category, as it can easily be detected by inspecting a single feature shown in Figure 3.1. Like SWaT, WADI contains redundant features, 33 in total, as well as another eight features with either missing or NaN values, bringing the total invalid feature count to 41 from the original 123, corresponding to almost a third. Such invalid features lead to ambiguity when benchmarking, as some researchers may choose to fill missing values and retain redundant features while others may omit the channels altogether, making the results incomparable across different papers and unnecessarily increasing complexity [33].

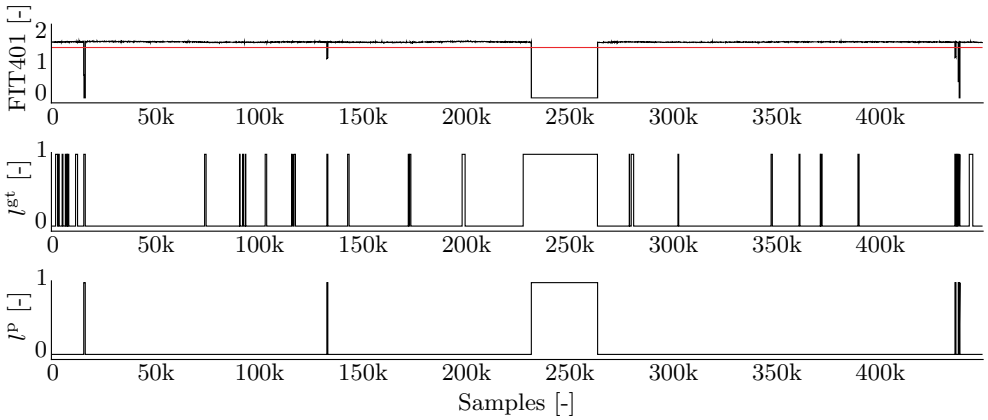

Figure 3.1: FIT401 channel, as well as ground-truth label vector l^{gt} and predicted label vector l^{p} from the SWaT dataset plotted.

The SMAP and the MSL [32] datasets are both published by NASA. They are divided into training and testing subsets, with the training data containing only nominal time steps, making them represent semi-supervised problems. According to the supplementary material by Wu and Keogh [31], both datasets exhibit issues such as triviality and an unrealistic density of anomalies. Additionally, while these datasets are technically multivariate, only the first channel (the telemetry channel) displays dynamic and anomalous behaviour. The remaining channels are binary, one-hot encoded command channels without any dynamic behaviour.

The SMD [2] dataset, consists of 28 training and test time series, respectively, each with 38 channels. At first sight, this data set may seem to present a discrete-sequence problem, however the data collected stems from different 28 server machines, which should be considered separately. This means that, for each server machine, there is a single training and test sequence, making this a continuous-sequence problem. Similar to SMAP and MSL, SMD suffers from triviality and an unrealistic anomaly density. This triviality is demonstrated by Wu and Keogh [31], where anomalies are detected using minimal code like the moving standard deviation.

As a solution to the benchmarking problems outlined in this subsection, Wu and Keogh [31] propose using a collection of 250 sequences, the Hexagon ML/UCR Time Series Anomaly Detection dataset. While addressing the identified flaws, this collection exclusively features univariate sequences, and hence it cannot be recommended as a realistic representation of complex real-world anomaly detection tasks.

Additionally, Wenig et al. [34] point out that the SWaT, WADI and SMD datasets feature mostly univariate anomalies, i.e. anomalous behaviour manifests itself in a

single channel. They found that, in these datasets, univariate approaches mostly outperform multivariate approaches due to the overwhelming presence of such univariate anomalies. However, once multivariate anomalies are injected, multivariate approaches outperform their univariate counterparts.

Following the work by Wu and Keogh [31], changes to datasets, new datasets and dataset generation frameworks are proposed with the goal of addressing the dataset flaws.

As part of the TimeEval framework, Wenig et al. [35] propose the good time series anomaly generator (GutenTAG), a tool for time series dataset generation, which can generate nominal and anomalous multivariate time series. It works by combining base oscillations, like sine or electrocardiogram-like waves, and injecting different types of pre-defined anomalies. However, GutenTAG only represents the tool to create a dataset, not the dataset itself. Further fragmentation of the research field will occur if no dataset resulting from GutenTAG is agreed upon, though GutenTAG-based data used in a benchmarking paper by the same authors [16] may serve as a reference. While the time series data generated by GutenTAG may be complex due to the different combinations of base oscillations, it still lacks the relationship to the real world. Arguably, the main purpose of research on time series anomaly detection is to solve real-world problems, and hence any evaluation should also consider real-world or real-world-inspired data.

Multivariate time series anomaly detection benchmark suites (mTADS) [36] is a collection of two types of datasets, one generated with GutenTAG and another based on simulation of the Lotka-Volterra equations which represent the relationship between one predator, two prey populations and another population that is both predator and prey. One interesting aspect of this dataset is that training data is offered both with and without anomalous behaviour, allowing for semi- and unsupervised anomaly detection. Despite being real world-inspired, the two equations are still fairly simple, yielding time series data with only four features.

All the data sets mentioned above are continuous-sequence problems, meaning there is a single test time series with both nominal and anomalous sub-sequences. As mentioned in Chapter 2.1, there are real-world applications which are discrete-sequence in nature. These applications may contain patterns in training data that are not present in test data but are considered nominal, nonetheless, requiring model-based approaches to be able to generalise. This aspect in particular is investigated more thoroughly in Chapter 5.3. Due to the discrete-sequence nature of some real-world problems, there is a need for such datasets to shed light on the viability of

approaches in addition to their performance in the above-mentioned datasets. As mentioned in Chapter 2.1, a key property of multivariate time series is the correlation between features, and hence datasets aiming to represent real-world problems should include multivariate anomalies.

3.1.2 Evaluation Metrics

Evaluation metrics allow different anomaly detection performance aspects to be quantified and compared. These metrics can generally be separated into two different classes: calibrated and uncalibrated. In short, calibrated metrics represent anomaly detection performance at a specific threshold, while uncalibrated metrics for a range of all possible thresholds. By representing the aggregate detection performance over all thresholds, the latter bypasses the threshold choice, though they do not denote the real-world detection performance in a productive scenario. To provide a consistent notation throughout the following subsections, the following notation system is introduced. Reconsider a testing subset $\mathcal{D}^{\text{test}}$ in the dataset $\mathcal{D}$ containing N sequences that can be entirely nominal, entirely anomalous, or partially anomalous, such that $\mathcal{D}^{\text{test}} = \{\mathcal{S}_1, ..., \mathcal{S}_n, ..., \mathcal{S}_N\}$, where $\mathcal{S}_n \in \mathbb{R}^{T_n \times d_{\mathcal{D}}}$. Here, T_n is the number of time steps in sequence $\mathcal{S}_n$, which varies depending on $\mathcal{S}_n$ and $d_{\mathcal{D}}$ the number of features, i.e. the dimensionality of $\mathcal{S}_n$. The ground-truth label set L^{gt} corresponding to $\mathcal{D}^{\text{test}}$ contains a binary label vector for each sequence, such that $L^{\text{gt}} = [\boldsymbol{l}_1^{\text{gt}}, ..., \boldsymbol{l}_n^{\text{gt}}, ..., \boldsymbol{l}_N^{\text{gt}}]$, where $\boldsymbol{l}_n^{\text{gt}} \in \mathbb{B}^{T_n}$. Anomalous time steps are assigned a 1 and nominal time steps a 0. Therefore, for a given nominal sequence $\sum_{t=0}^{T_n} l_{n,t}^{\text{gt}} = 0$, for an entirely anomalous sequence $\sum_{t=0}^{T_n} l_{n,t}^{\text{gt}} = T_n$ and for a partially anomalous sequence $0 < \sum_{t=0}^{T_n} l_{n,t}^{\text{gt}} < T_n$. As a counterpart to the ground-truth label set L^{gt}, there is the corresponding predicted label set L^{p}, such that $L^{\text{p}} = [\boldsymbol{l}_1^{\text{p}}, ..., \boldsymbol{l}_n^{\text{p}}, ..., \boldsymbol{l}_N^{\text{p}}]$, where $\boldsymbol{l}_n^{\text{p}} \in \mathbb{B}^{T_n}$.

Calibrated Metrics

Similar to binary classification, anomaly detection generally segregates two classes using a threshold, except that one class is far rarer than the other, and hence can be considered as imbalanced classification. This threshold is often applied to some sort of anomaly score, usually a function of time, which is usually closely related to the loss function in model-based approaches. Correct predictions can then be labelled as true positives and true negatives, and incorrect ones into false positives and false negatives, where negative refers to a nominal case and positive to an anomalous case. From here on, the number of true positives, i.e. the number of correctly identified anomalies, is

represented by $N_{\text{tp}} \in \mathbb{N}$, while the number of true negatives, i.e. the number of correctly identified nominal time steps, is described by $N_{\text{tn}} \in \mathbb{N}$. The number of false positives, i.e. the number of nominal time steps incorrectly identified as anomalies, is depicted by $N_{\text{fp}} \in \mathbb{N}$ and lastly, the number of false negatives, i.e. the number of anomalous time steps incorrectly identified as nominal, is expressed by $N_{\text{fn}} \in \mathbb{N}$. For now, label vector temporality is ignored, and each time step is considered individually, and therefore all anomalous time steps are treated as individual point anomalies. Depending on the match between the predicted label vector $\boldsymbol{l}_n^{\text{p}}$ and the ground-truth label vector $\boldsymbol{l}_n^{\text{gt}}$, the number of true positives, true negatives, false negatives and true positives can be obtained; see Equation 3.1.

$$\begin{aligned} N_{\text{tp}} &= \sum_{n=1}^{N} \boldsymbol{l}_n^{\text{p}} \cdot \boldsymbol{l}_n^{\text{gt}} \qquad & N_{\text{tn}} &= \sum_{n=1}^{N} (\boldsymbol{o}_n - \boldsymbol{l}_n^{\text{p}}) \cdot (\boldsymbol{o}_n - \boldsymbol{l}_n^{\text{gt}}) \\ N_{\text{fn}} &= \sum_{n=1}^{N} (\boldsymbol{o}_n - \boldsymbol{l}_n^{\text{p}}) \cdot \boldsymbol{l}_n^{\text{gt}} \qquad & N_{\text{fp}} &= \sum_{n=1}^{N} \boldsymbol{l}_n^{\text{p}} \cdot (\boldsymbol{o}_n - \boldsymbol{l}_n^{\text{gt}}) \end{aligned} \tag{3.1}$$

Here, $\boldsymbol{o}_n$ is a vector of ones the same size as $\boldsymbol{l}_n^{\text{gt}}$ and $\boldsymbol{l}_n^{\text{p}}$.

Precision P is a metric that shows the number of true positives in relation to the total number of detections, i.e. how many of the predicted positive time steps are correct, and can be calculated using Equation 3.2. As is evident, a smaller number of false positives leads to a higher precision.

$$P = \frac{N_{\text{tp}}}{N_{\text{tp}} + N_{\text{fp}}} \tag{3.2}$$

Recall R is often presented in combination with precision, as it measures the number of true positives in relation to the total ground-truth anomaly count, i.e. how many of the anomalous time steps are detected, and can be calculated using Equation 3.3. As is evident, a smaller number of false negatives leads to a higher recall.

$$R = \frac{N_{\text{tp}}}{N_{\text{tp}} + N_{\text{fn}}} \tag{3.3}$$

Precision and recall can be summarised into a single metric, the F_β score, which represents the weighted harmonic mean of the two metrics, as shown in Equation 3.4.

$$F_\beta = \frac{(1+\beta^2) \cdot N_{\text{tp}}}{(1+\beta^2) \cdot N_{\text{tp}} + \beta^2 \cdot N_{\text{fn}} + N_{\text{fp}}} = (1+\beta^2) \cdot \frac{P \cdot R}{(\beta^2 \cdot P) + R} \tag{3.4}$$

Typically, precision and recall are equally weighted, hence $\beta = 1$, which is referred to the F_1 score; see Equation 3.5.

$$F_1 = \frac{N_{\text{tp}}}{N_{\text{tp}} + \frac{1}{2} \cdot (N_{\text{fn}} + N_{\text{fp}})} = 2 \cdot \frac{P \cdot R}{P + R} \tag{3.5}$$

The metrics in Equations 3.1–3.5 are referred to as *calibrated metrics*, since they represent anomaly detection performance at a specific threshold/sensitivity.

One metric that is often used in binary classification is the accuracy Φ, which represents the total number of correct classifications in relation to all samples, as shown in Equation 3.6.

$$\Phi = \frac{N_{\text{tp}} + N_{\text{tn}}}{N_{\text{tp}} + N_{\text{fn}} + N_{\text{fp}} + N_{\text{tn}}} \tag{3.6}$$

However, accuracy is unsuitable for anomaly detection, given the imbalanced nature of the problem. For instance, consider a scenario with $N_{\text{n}} = 90$ ground-truth nominal and $N_{\text{a}} = 10$ ground-truth anomalous time steps, where $N_{\text{n}} = N_{\text{tn}} + N_{\text{fp}}$ and $N_{\text{n}} = N_{\text{fn}} + N_{\text{tp}}$. An anomaly detector that classifies all 100 time steps as nominal would have an accuracy figure of $\Phi = 0.9$, despite it detecting no anomalies at all.

One alternative to regular accuracy is the balanced accuracy Φ_B, which calculates the accuracy in both classes separately and then takes the average, as in Equation 3.7.

$$\Phi_B = \frac{1}{2} \left(\frac{N_{\text{tp}}}{N_{\text{tp}} + N_{\text{fn}}} + \frac{N_{\text{tn}}}{N_{\text{tn}} + N_{\text{fp}}} \right) \tag{3.7}$$

While more suitable for anomaly detection than "regular" accuracy, this performance metric can be still skewed by the natural imbalance between nominal and anomalous samples, which could be a reason why it has seen no mention in any of the literature surveyed in this work. To illustrate this, consider the same case as above, where $N_{\text{n}} = 90$ and $N_{\text{a}} = 10$ and an arbitrary anomaly detector marks all samples as nominal, i.e. $N_{\text{tn}} = 90$, $N_{\text{fn}} = 10$ and $N_{\text{tp}} = N_{\text{fp}} = 0$. This anomaly detector would then still be assigned a balanced accuracy of $\Phi_B = 0.5$, despite its failure of detecting any anomalies. In contrast, such an anomaly detector would obtain an F_1 score of 0, an arguably a more intuitive result.

As previously mentioned, these metrics are generally used in imbalanced classification and are sample-based, i.e. each time step is considered individually.

The first proposal for metrics apt to evaluate sub-sequence anomalies is given by Xu et al. [37]. They apply the metrics such that a ground-truth anomalous subsequence is considered a true positive if it overlaps with any predicted anomalous time

step. False positives and false negatives are obtained as previously. This method is generally referred to as point-adjustment. Despite its prominence in the literature, point-adjusted metrics come with a number of problems. Firstly, in a case where only the last anomalous time step is predicted positive, a significant delay in the detection is present which can be undesired. Furthermore, it makes anomaly detection trivial. In the case of a single long sub-sequence anomaly or a whole-sequence anomaly, a single predicted positive time step within it would lead to true positives for all ground-truth positive time steps within the sub-sequence anomaly, despite all other time steps not being classified correctly. To further underline the effect of point-adjustment on metrics, consider Figure 3.2. Without point-adjustment, the figure shows $N_{\text{tp}} = 4$,

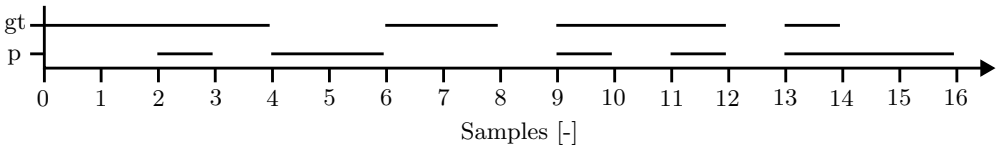

Figure 3.2: Plot of arbitrary ground-truth and predicted label vectors $\boldsymbol{l}_n^{\text{gt}}$ and $\boldsymbol{l}_n^{\text{p}}$, respectively, where the presence of the line indicates anomalous time steps. The anomalous subsequences occupy the intervals $[0, 4)$, $[6, 8)$, $[9, 12)$, $[13, 14)$, respectively. The predicted positive subsequences occupy the intervals $[2, 3)$, $[4, 6)$, $= [9, 10)$, $[11, 12)$, $[13, 16)$, respectively.

$N_{\text{fp}} = 4$, $N_{\text{tn}} = 2$ and $N_{\text{fn}} = 6$, and hence a precision of 0.50, a recall of 0.40 and therefore an F_1 score of 0.44. With point-adjustment, however, the figure shows $N_{\text{tp}} = 8$, $N_{\text{fp}} = 4$, $N_{\text{tn}} = 2$ and $N_{\text{fn}} = 2$, and hence a precision of 0.66 and a recall of 0.80 and therefore an F_1 score of 0.72. Clearly, point-adjusted metrics are not well suited for sub-sequence anomaly detection, as they are too forgiving and ignore delays in detection. In fact, Doshi et al. [38] show that sampling a uniform distribution parameterised with an alarm probability can outperform most state-of-the-art models when point-adjusted metrics are used.

Tatbul et al. [39] propose so-called range-based recall R_{rb} and precision P_{rb}. For the range-based recall R_{rb} calculation, four aspects are relevant to anomalous subsequences within a time series. The first is existence, which is analogous to the point-adjusted way of marking a true positive. The corresponding existence score ϵ for each ground-truth anomaly is 1 if any predicted positive time steps overlap with it and 0 otherwise. In the context of Figure 3.2 the existence score is 1 for the first, third and fourth ground-truth sub-sequence anomaly and 0 for the second one. Then, there is the position, i.e. what part of the predicted anomaly overlaps with the ground-truth anomaly. For this aspect, this method requires an application-related positional bias ζ.

There may be use-cases where correctly predicting certain sections as positive within the ground-truth anomaly and, therefore, the front, back, middle, or all time steps should be emphasised. The next aspect is size, i.e. how close the size of the predicted anomaly is to the ground-truth anomaly, denoted by the overlap size ω, which is a ratio between the number of predicted and ground-truth positive time steps within the anomaly. In Figure 3.2 the overlap size is low for the first ground-truth anomalous sub-sequence, higher for the third one and 1 for the fourth one. For the second ground-truth anomaly, the overlap size is undefined, as there is no predicted time step that overlaps with it. The overlap size is also a function of the aforementioned application-related positional bias ζ. Lastly, there is cardinality. One example of a cardinality score γ would be one that indicates how many predicted positive sub-sequences overlap with each ground-truth positive sub-sequence. This is especially useful to punish approaches that predict several positive sub-sequences within a single ground-truth positive sub-sequence rather than a single continuous one. In Figure 3.2 the cardinality score for the first and fourth ground-truth anomalous sub-sequence would be higher than for the third one. Again, for the second one, this would be undefined due to the lack of corresponding predicted positive time steps. The cardinality factor is used to scale the overlap size figure, yielding the overlap score which is combined with the existence score using a pre-defined weight α to yield the range-based recall for each ground-truth anomaly i, as shown in Equation 3.8.

$$R_{\text{rb},i} = \alpha \cdot \epsilon + (1 - \alpha) \cdot \gamma \cdot \omega \tag{3.8}$$

For the range-based precision P_{rb} the existence score ϵ does not apply and hence only the overlap size ω, positional bias δ, and cardinality score γ are considered for each predicted anomaly. The aforementioned scores are then applied inversely to the above, i.e. for each predicted anomaly j, rather than for each ground-truth anomaly i. Finally, the individual range-based recalls and precisions are averaged for all ground-truth anomalies and all predicted anomalies to obtain the final range-based recall R_{rb} and precision P_{rb}, respectively. While this way of evaluating anomaly detection performance is fairly robust, it assumes that in a test time series, there is always a predicted anomaly, which is not always the case in discrete-sequence problems. In a case where this does not happen, the method runs into a numerical error. Furthermore, it requires the setting of bias γ, as well as weight α, which sets the balance between existence and size, position, and cardinality.

Hwang et al. [40] propose an extension to the point-adjusted metrics, called the

time-aware precision P_{ta} and time-aware recall R_{ta}. For their metrics, Hwang et al. [40] assume that each positive ground-truth has an ambiguous tail of length ψ, where ψ time steps after the end of a ground-truth anomaly are considered anomalous too. Both time-aware precision P_{ta} and time-aware recall R_{ta} consist of two components, the detection score and the portion score. The detection score components of the time-aware precision and recall are P^{d}_{ta} and R^{d}_{ta}, respectively. The way true positives are counted is similar to the point-adjusted score, except that it requires the predicted anomaly to overlap with the ground truth by a set threshold τ. The portion score components P^{p}_{ta} and R^{p}_{ta} are defined as the average overlap between ground-truth and predicted anomalies. The final time-aware precision P_{ta} and recall R_{ta} is obtained using the weighted sum of the detection and portion scores, respectively, parameterised by weight α. Like range-based recall R_{rb} and precision P_{rb}, time-aware precision P_{ta} and recall R_{ta}, α needs to be manually defined.

Garg et al. [41] introduce a different evolution to point-adjusted metrics [37] by calculating the time-wise precision P_{tw} and the event-wise recall R_{ew}. Both are calculated using Equations 3.2 and 3.3, with the main difference being how N_{tp}, N_{fp} and N_{fn} are counted. For the time-wise precision P_{tw}, true positives and false positives are counted as shown in Equation 3.1, i.e. labels are counted based on time steps. For the event-wise recall R_{ew}, however, true positives are counted in a point-adjusted way [37], while false negatives are simply the number of anomalous sub-sequences where no time steps are labelled anomalous. Using the time-wise precision P_{tw} and the event-wise recall R_{ew}, the so-called composite F-score Fc_1 can be calculated using Equation 3.5. For a straight-forward example, consider Figure 3.2. Time-wise precision P_{tw} is 0.5, since $N_{\text{tp,tw}} = 4$ and $N_{\text{fp,tw}} = 4$, while event-wise recall R_{ew} is 0.75, since $N_{\text{tp,ew}} = 3$ and $N_{\text{fn,ew}} = 1$, which leads to a composite F-score of $Fc_1 = 0.6$. Though applicable to discrete-sequence problems, this set of metrics does not account for the timeliness of detections. An approach that tends to detect a single time step at the end of each anomalous subsequence would be indistinguishable from another that is very timely by predicting only the first time step as positive or that predicts all ground-truth time steps as positive. Additionally, this set of metrics do not take predictive patterns into account, meaning that an approach flagging every other time step within an anomalous subsequence is indistinguishable from another that perfectly flags all time steps.

To specifically address the delay in detection, Doshi et al. [38] propose a metric called the average detection delay $\bar{D}$, which is the mean delay for an approach to label a ground-truth anomaly as such over all ground-truth anomalies. For example, the detection delay D for the first ground-truth anomaly in Figure 3.2 would be two time

steps and zero for the third and fourth ground-truth anomalies. To cap the delay time of detections, a maximum tolerable delay δ_{max} is used if the detection takes longer than δ_{max}, therefore the worst $\bar{D}$ score possible is $\bar{D} = \delta_{max}$. The average detection delay $\bar{D}$ can be normalised using δ_{max}, in which case the worst value is $\tilde{D} = \bar{D}/\delta_{max} = 1$. It is undesirable to flag time steps too early, i.e. before the first ground-truth positive time step, hence Doshi et al. [38] propose the sequence alarm precision P_{sa}, where a false positive is recorded for premature detections. It is calculated the same way regular precision is, except that a false positive is defined as an early flag, therefore P_{sa} is the number of early flags relative to all flags. Like the two proposed metrics discussed previously, the manual setting of δ_{max} is required. Furthermore, it is unclear what the delay should be if there is no predicted anomaly in a given test sequence. Also, Doshi et al. [38] do not propose a calibrated metric that combines $\tilde{D}$ and P_{sa}, like the F_1 score combines precision P and recall R.

In an attempt to create a set of interpretable and parameter-free metrics, Huet et al. [42] propose the following. First, a local affiliation between the ground-truth label vector $\boldsymbol{l}_n^{\text{gt}}$ and the predicted label vector $\boldsymbol{l}_n^{\text{p}}$ is established. This is done by splitting test sequences into chunks, each with exactly one ground-truth anomaly. For the precision in each chunk, they calculate the average temporal distance d_P between any false positive time steps and the closest ground-truth positive time step. When this average temporal distance is zero, it means that no time steps within the chunk are false positives. In the case of the recall, this is flipped, so that the average temporal distances d_R between any false negative time steps and the closest ground-truth time step are calculated. When this average temporal distance is zero, it means that no time steps within the chunk are false negatives. Then, the average directed distances are averaged over all chunks, resulting in P_{td} and R_{td}, since the temporal distances are interpreted as precision and recall. To provide context on the anomaly detection performance, the absolute temporal distances can be replaced with sample-based precision and recall probabilities p_P and p_R, respectively. These are normalised against random predictions, hence a value of above 0.5 in either means that they are better than random predictions. Like the metrics proposed previously, Huet et al. [42] do not address cases where no predicted anomalies are present in a chunk.

Wagner et al. [33] propose several improvements to the metrics introduced by Tatbul et al. [39]. The first adjustment involves the weight parameter α. They point out that the existence score is unnecessary as the other scores already imply the existence, i.e. overlap of the ground truth and predicted positive sub-sequence, hence α is set to 0. Secondly, they suggest a new cardinality score that is consistent across

all types of bias; however, they point out the importance of the constant bias due to its neutrality. Lastly, Wagner et al. [33] change the way the individual precision scores are combined. Tatbul et al. [39] simply take the mean of all precisions, but rather than dividing the sum of precisions by the number of predictions, Wagner et al. [33] use a weighted sum of precisions, which depends on the length of the respective prediction. These improvements lead to a parameter-free evolution of the range-based precision and recall, at least when constant bias is considered the default, however, the new metrics still suffer from the same assumption that for every ground-truth anomaly, there is at least one predicted anomaly.

To avoid ambiguity on how multiple sub-sequence anomalies within a sequence should be detected and counted, Wu and Keogh [31] suggest framing the anomaly detection problem such that each test sequence has a single anomaly that is either detected or not detected. Datasets with a single test time series containing multiple anomalies would require splicing to create multiple sub-sequences, each with a single anomaly.

Table 3.2 summarises the key distinguishing factors of the metrics introduced before. As is evident, almost all methods are compatible with all types of anomalies, with

Table 3.2: Table summarising the key weaknesses and strengths of the metrics proposed in this subsection. The key is as follows, PA: point anomaly, SSA: sub-sequence anomaly, WSA: whole-sequence anomaly, CS: continuous-sequence anomaly detection, DS: discrete-sequence anomaly detection, PF: parameter-free, ADD: aware of detection delay

Publications	PA	SSA	WSA	CS	DS	PF	ADD
Xu et al. [37]	✓	✓	✓	✓	✓	✓	
Tatbul et al. [39]	✓	✓	✓	✓			✓
Hwang et al. [40]	✓	✓	✓	✓			
Garg et al. [41]	✓	✓	✓	✓	✓	✓	
Doshi et al. [38]	✓	✓		✓			✓
Huet et al. [42]	✓	✓	✓	✓		✓	✓
Wagner et al. [33]	✓	✓	✓	✓		✓	✓

the sequence alarm precision P_{sa} by Doshi et al. [38] being the exception. Remember that P_{sa} represents the number of early flags relative to all flags. In the case of time series anomalies, this cannot be applied, since it is impossible to flag a time step early. All metrics are introduced in the context of continuous-sequence anomaly detection, and therefore it is nowhere specified how they can be extended to discrete-sequence anomaly detection. Some research [33, 37, 42] offers non-parametric solutions, which are preferred for benchmarking, as it removes tuneable elements that can alter results

and hinder fair comparison. Lastly, all metrics but the point-adjusted ones [37] require an existing prediction for each ground-truth anomaly, else they return numerical errors. This is especially relevant in discrete-sequence anomaly detection, where a whole-sequence anomaly may be entirely labelled as nominal. Despite the extensive work done in the area, there is still a clear need for a set of metrics that take timeliness of detections into account, are parameter-free and do not assume predicted anomalies within every test time series, as is the case in discrete-sequence problems.

Uncalibrated Metrics

Metrics that do not require a specific threshold are referred to as uncalibrated metrics. Rather than representing anomaly detection performance at a certain threshold, they depict anomaly detection performance over a range of possible thresholds.

The most prominent metric is the area under the receiver operating characteristic curve A_{ROC}, which can be obtained using a variety of discrete integration methods, such as the midpoint rule, Simpson's rule or the trapezoidal rule. The A_{ROC} is a function of the true-positive rate, or in other words, recall R, w.r.t. the false-positive rate Q and hence, in the case of the trapezoidal rule, the discrete A_{ROC} is obtained using Equation 3.9.

$$A_{\text{ROC}} = \frac{1}{2}\sum_{k=1}^{K}(R_{k-1} + R_k) \cdot \Delta Q_k \tag{3.9}$$

Here, the false-positive rate Q is computed as shown in Equation 3.10, k is the index of discrete values and $\Delta Q_k = Q_k - Q_{k-1}$.

$$Q = \frac{N_{\text{fp}}}{N_{\text{fp}} + N_{\text{tn}}} \tag{3.10}$$

Similar to accuracy Φ, A_{ROC} is unsuitable for imbalanced classification and therefore anomaly detection, as in the same example the false positive rate is disproportionally influenced by the majority class, i.e. the nominal class.

Another metric is the area under the precision-recall curve A_{PR}. The precision-recall curve is a function of precision w.r.t. recall and hence, the discrete area under the precision-recall curve is obtained using Equation 3.11.

$$A_{\text{PR}} = \frac{1}{2}\sum_{k=1}^{K}(P_{k-1} + P_k) \cdot \Delta R_k \tag{3.11}$$

Here, k is the index of discrete values. An example of a precision-recall curve is shown

in Figure 3.3.

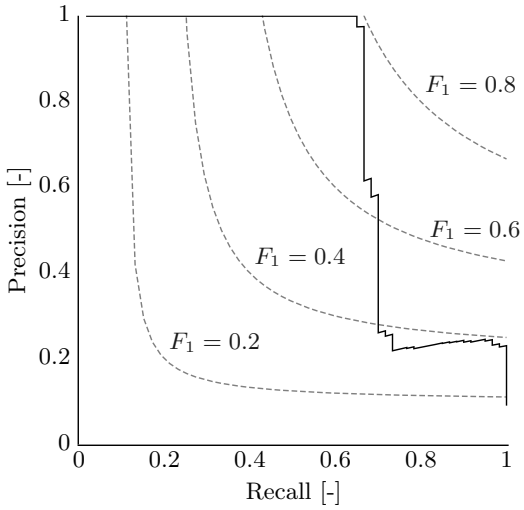

Figure 3.3: Example of the precision-recall curve for an unspecified approach. When calibrated, the detection performance of the unspecified approach would be located on a single point along this curve. The closer it is to the $P = R = 1$ point, the higher the best possible F_1 score. Dashed lines denote paths along which the respective F_1 score is achievable.

The average precision $\bar{P}$ can be seen as an alternative method of reducing a precision-recall curve to a single scalar. It is calculated as the weighted mean precision over all values along the precision-recall curve, as shown in Equation 3.12. As far as the author is aware, only Audibert et al. [43] use this metric for evaluating anomaly detection approaches.

$$\bar{P} = \sum_{k=1}^{K} P_k \cdot \Delta R_k \tag{3.12}$$

In essence, this is just another discrete way of integrating the precision-recall curve. The average precision metric therefore simply denotes the A_{PR} when right-endpoint integration is used, rather than the trapezoidal integration rule.

Despite Tatbul et al. [39] not explicitly discussing it, the area under the range-based precision and recall curve $A_{\mathrm{P_rR_r}}$ can also be calculated. Schmidl et al. [16] first use it, while Wagner et al. [33] later apply it to their improved range-based precision and recall, which aim to provide scoring metrics apt for sub-sequence anomalies in addition to point anomalies.

Doshi et al. [38] suggest combining both the normalised average detection delay $\tilde{D}$

and sequence alarm precision P_{sa} by obtaining the area under the P_{sa}-$\tilde{D}$ curve $A_{P_{\text{sa}}\tilde{D}}$, similar to the A_{PR}.

Paparrizos et al. [44] propose a new way to count true positives, true negatives, false positives and false negatives which, like Hwang et al. [40], accounts for label uncertainty. In their work, they propose transforming the binary label vector into a continuous one. First, they split the test time series into chunks like Huet et al. [42], each with a single ground-truth anomaly within. Then, Paparrizos et al. [44] extend each ground-truth sub-sequence by adding an ambiguous tail to the end of the sub-sequence, like Hwang et al. [40], and an ambiguous head to the beginning, both of length ψ. The values inside the ambiguous head and tail are not binary, however, but are assigned a gradient. When two sub-sequence anomalies are less than ψ apart, the tail and the head, respectively, are superimposed. Moreover, the length of the ambiguous head and tail ψ can be set to a variety of values: half the period of the sequence, the average length of the sub-sequence anomalies or a manually defined value. The metrics in this subsection are obtained the same way as shown in Equation 3.1, i.e. sampled-based, however, as is evident, these are no longer whole numbers but instead rational numbers, i.e. $N_{\text{tp}} \in \mathbb{Q}$, $N_{\text{fp}} \in \mathbb{Q}$, $N_{\text{tn}} \in \mathbb{Q}$ and $N_{\text{fn}} \in \mathbb{Q}$. While this reduces interpretability, precision and recall can still be calculated as shown in Equations 3.2 and 3.3. Furthermore, the sum of these metrics still equates to the number of samples in the ground-truth label vector $\boldsymbol{l}_n^{\text{gt}}$. This counting system is not a good match for the A_{PR} and A_{ROC} metrics as the period of the sequence and the average length of the sub-sequence anomalies are not known a priori; therefore, ψ will default to a manually defined value, turning this set of metrics into essentially parametric. To circumvent this issue, Paparrizos et al. [44] propose two volume metrics: the volume under the receiver operating characteristic surface V_{ROC} and under the precision-recall surface V_{PR}. V_{ROC} and V_{PR} are calculated similarly to their area-based counterparts. In essence, they are an extension that takes a variable ambiguous head and tail length ψ into account, turning a 2D curve into a 3D surface. The volume under the surfaces is therefore found by integrating across two axes, rather than just one. The discrete integration for the two surface metrics are shown in Equations 3.13 and 3.14.

$$V_{\text{ROC}} = \frac{1}{4} \sum_{\psi=1}^{\Psi} \sum_{k=1}^{K} (R_{\psi,k-1} + R_{\psi,k}) \cdot \Delta Q_{\psi,k} \tag{3.13}$$

$$V_{\text{PR}} = \frac{1}{4} \sum_{\psi=1}^{\Psi} \sum_{k=1}^{K} (P_{\psi,k-1} + P_{\psi,k}) \cdot \Delta R_{\psi,k} \tag{3.14}$$

Here, k is the index of discrete values and ψ the ambiguous head and tail length of anomalies.

Like with the calibrated metrics, there is no uncalibrated metric that represents anomaly detection performance for a range of thresholds, while also being compatible with all anomaly types, non-parametric and non-assuming of positive predictions.

3.1.3 End-to-end Benchmarking Frameworks

To further streamline the standardised comparison between different approaches, a number of end-to-end benchmarking frameworks are proposed as well. They contain different datasets, a set of metrics, and implementations of various anomaly detection approaches.

Wenig et al. [35] published the first effort in this area with *TimeEval*. TimeEval features a multitude of univariate and multivariate data sets, and its biggest strength is the inclusion of the GutenTAG dataset generation tool. One standout feature of TimeEval is the large number of containerised anomaly detection approaches provided, which allows for high level of control over the environment and computing resources allocated to anomaly detection approaches, though it lacks support GPU-accelerated training for approaches that require it.

Seemingly independently and simultaneously, Wagner et al. [33] propose *TimeSeAD*. It contains a more limited dataset selection than Wenig et al. [35], but including with a modified SMD dataset, where problematic sequences are removed. TimeSeAD also provides the parameter-free evolution of the range-based recall R_{rb} and precision P_{rb}.

Liu and Paparrizos [45] propose another benchmarking framework called the *TSB-AD*, which mainly contributes to the number of data sets included, as well as the inclusion of one multivariate time series foundation model, i.e. a model pre-trained on time series data. Additionally, TSB-AD allows for the calculation of the volume under the surface metrics V_{ROC} and V_{PR} proposed by Paparrizos et al. [44].

While the initial efforts to standardise benchmarking are commendable, new frameworks try to compete by offering an increasing number of datasets and anomaly detection implementations. Meanwhile, TimeEval, as the oldest framework, has continually been receiving code contributions years after launch, particularly in areas such as metrics and implemented approaches, further contributing to a fragmented benchmarking landscape. For instance, both the improved range-based recall and precision [33] and the volume under surface metrics [44] were added since its inception. While each

framework brings valuable and unique features, the research community would benefit more from a widely accepted, collaborative framework rather than individual competing initiatives. An ideal scenario would be one where new advancements in the research field are directly integrated into the source code of a widely accepted framework.

Aside from fragmentation, current frameworks also lack the compatibility with discrete-sequence datasets, due to their absence from the public domain and the lack of robust evaluation metrics compatible with such problems.

3.2 Navigating Through the Time Series Anomaly Detection Landscape

Over the years, a multitude of model-based approaches are proposed to tackle the detection of anomalies in multivariate time series. They can be largely segmented into four different categories: predictive, reconstructive, generative and transformer-based approaches.

Predictive models are trained to predict a time horizon given a finite number of past values. Consider a time series window $\mathcal{W} \in \mathbb{R}^{w \times d_{\mathcal{D}}}$ and a predictive model $\mathcal{M}$. In most publications, $\mathcal{M}$ is trained on a prediction horizon of h time steps given $w-h$ past values, as shown in Equation 3.15.

$$\hat{\mathcal{W}}_{w-h+1:w} = \mathcal{M}(\mathcal{W}_{1:w-h}) \tag{3.15}$$

Such predictive models [32, 46–56] can be used for anomaly detection by comparing their predictions with the observed value using a variety of techniques to obtain an error metric. The key assumption is that nominal, i.e. anomaly-free, sequences are predicted with a smaller error than anomalies.

Reconstructive modelling is usually based on some variation of an autoencoder. Autoencoders consist of an encoder $\mathcal{M}_{\text{Enc}}$ and decoder $\mathcal{M}_{\text{Dec}}$, which generally mirror each other in architecture. The encoder compresses a given input time series window $\mathcal{W} \in \mathbb{R}^{w \times d_{\mathcal{D}}}$ into a latent vector $\mathbf{z} \in \mathbb{R}^{u \times d_{\mathbf{z}}}$, i.e. a reduced representation of the input space. Depending on the approach, this latent representation $\mathbf{z}$ may or may not contain temporality, i.e. $u > 1$ or $u = 1$, respectively. The latent vector is then expanded again to reconstruct the input $\mathcal{W}$ through the decoder, as shown in Equation 3.16.

$$\begin{aligned} \mathbf{z} &= \mathcal{M}_{\text{Enc}}(\mathcal{W}) \\ \hat{\mathcal{W}} &= \mathcal{M}_{\text{Dec}}(\mathbf{z}) \end{aligned} \tag{3.16}$$

Reconstructive models [57–73] rely on the assumption that the reconstruction error will be large when faced with anomalous data.

Generative models can be segmented into two model types: variational autoencoders (VAEs) [20, 21] and generative adversarial networks (GANs) [22]. The former bears some similarity to traditional autoencoders, apart from the fact that the low-dimensionality representation is not mapped to a vector $\mathbf{z}$ but rather a distribution, usually a Gaussian distribution with mean and standard deviation parameters $(\boldsymbol{\mu}_{\mathbf{z}}, \boldsymbol{\sigma}_{\mathbf{z}})$ from which a latent vector $\mathbf{z} \in \mathbb{R}^{u \times d_{\mathbf{z}}}$ is sampled; see Equation 3.17.

$$\begin{aligned} (\mu_{\mathbf{z}}, \sigma_{\mathbf{z}}) &= \mathcal{M}_{\text{Enc}}(\mathcal{W}) \\ \mathbf{z} &\sim (\boldsymbol{\mu}_{\mathbf{z}}, \boldsymbol{\sigma}_{\mathbf{z}}) \\ \hat{\mathcal{W}} &= \mathcal{M}_{\text{Dec}}(\mathbf{z}) \end{aligned} \tag{3.17}$$

Again, depending on the publication, the latent vector may or may not contain temporality. In some cases, the output of the decoder is not deterministic, but instead a distribution with parameters $(\boldsymbol{\mu}_{\mathcal{W}}, \boldsymbol{\sigma}_{\mathcal{W}})$ that approximates the input $\mathcal{W}$. In the generative modelling literature, the encoder of an autoencoder is also referred to as the recognition model and the decoder as the generative model. GANs, on the other hand, work by training a generator $\mathcal{M}_{\text{Gen}}$ network to generate a plausible time series window $\hat{\mathcal{W}} \in \mathbb{R}^{w \times d_{\mathcal{D}}}$ based on a sample from a normal distribution $\mathcal{N}(0, 1)$. A second model $\mathcal{M}_{\text{Disc}}$, the discriminator, then tries to classify the generated input window $\hat{\mathcal{W}} \in \mathbb{R}^{w \times d_{\mathcal{D}}}$ and the training example $\mathcal{W}$ by assigning a binary label $l \in \mathbb{B}$; see Equation 3.18.

$$\begin{aligned} \mathbf{z} &\sim \mathcal{N}(0, 1) \\ \hat{\mathcal{W}} &= \mathcal{M}_{\text{Gen}}(\mathbf{z}) \\ l &= \mathcal{M}_{\text{Disc}}(\mathcal{W}) \\ l &= \mathcal{M}_{\text{Disc}}(\hat{\mathcal{W}}) \end{aligned} \tag{3.18}$$

As training progresses, the generator becomes better at generating samples, hence discriminating between the generated input and the real data becomes harder. However, the discriminator also becomes better at telling real samples apart from generated samples. Hence, training can be thought of as a two-player min-max game [22]. Approaches based on generative models include [2, 74–87].

Transformers, first introduced by Vaswani et al. [23], are on the rise in the deep learning research landscape, thanks to significant advances enabled in natural language

processing by large language models. Transformer models have started to find their place in time series anomaly detection, though by far to a lesser extent than the previously discussed model families. The original transformer shows some resemblance to encoder-decoder architectures, though with a variety of improvements. First, it does not use recurrent but rather feed-forward layers to process input information to accelerate training through parallelisation, which is otherwise a sequential operation with long short-term memory (LSTM) and gated recurrent unit (GRU) layers. The temporal order of input data is maintained through the positional encoding of the inputs before entering the model. Furthermore, transformers use an evolution of the attention mechanism, called multi-headed attention, which allows the model to attend to information more effectively. A typical transformer block contains a feed-forward layer as well as a multi-headed attention layer, transformers can consist of several encoder and decoder blocks. Approaches based on transformer models include [38, 88–93].

As outlined in Chapter 3.1, flawed data sets have unfortunately found widespread adoption in the field of time series anomaly detection. Aside from flawed publicly available data sets, numerous publications use solely proprietary data sets [47, 58, 60, 63, 74, 80, 84, 87], which, while valid due to the flaws discussed present in public data sets, hinders direct comparison with other approaches and reproducibility. Some public data sets do not provide pre-determined training and testing subsets, and hence they are likely to be split differently in any publication using them, preventing comparison between approaches. Furthermore, the data sets that do provide training and testing splits, are also vague as to what exactly the training subset is composed of. Not providing labels for each sequence in the training subset is consistent with the unsupervised problem setting, but information on the contamination level, i.e. whether anomalies are present in the subset, is often not provided. Lastly, some publications only use a subset of a given data set, which further prevents a fair comparison, especially when performance metrics are provided as an average over all subsets.

As discussed in Chapter 3.1, multiple evaluation metrics apt for time series anomaly detection are proposed in literature, though none have proved to be accepted as the way forward in literature. Without reproducing the experiments in each paper, it is essentially impossible to directly compare the anomaly detection performance across the approaches reviewed above. Firstly, some research presents their results solely as an A_{ROC} figure, which, as specified in Chapter 3.1, is unsuitable for anomaly detection evaluation, while only two publications [67, 81] use the more appropriate A_{PR} metric.

Secondly, very few papers use both calibrated and uncalibrated metrics to denote anomaly detection performance. The two types of scores denote different aspects and are incompatible with each other, hence comparing work that only uses one of the metrics is not possible. Then, some publications use different types of F scores. The most used version is the F_1 score, where precision and recall are equally weighted; however, some publications use the $F_{0.5}$ score [32], the $F_{0.1}$ score [49, 50] or even the $F_{0.05}$ score [61] which all favour precision over recall. For obvious reasons, this makes a comparison of the approaches difficult, though this is a rare pattern throughout the literature and exclusively appears in older publications. Even when considering the papers that included F_1 scores, the numbers are mostly incomparable. Some publications calculate the point-wise F_1 score, while others use the point-adjusted F_1 score. Besides the fact that the point-adjusted F_1 score inflates evaluation metrics and makes the approaches look better than they actually are, it is incomparable to the point-wise F_1 score.

Lastly, there is disagreement as to what the concept of supervision means in anomaly detection. Recall Chapter 2.2, where unsupervised anomaly detection is defined as a problem independent of any labels. Clearly, that only applies to the training and validation subsets for model-based approaches, but to properly assess the anomaly detection performance, a labelled test subset is required. Quantifying the detection performance on test data allows practitioners to gauge how an approach would perform on a real-world problem, where labelled data is not available, and hence they should not use test data for calibrating approaches. Unfortunately, most of the literature uses labelled data, making any associated approach no longer truly unsupervised. The threshold choice is perhaps the single most important hyperparameter of an anomaly detection algorithm, as it defines the separation between predicted nominal and predicted anomalous time steps, though labelled data has very often been used to set it [32, 38, 48–51, 53–57, 59–62, 65, 66, 68, 69, 71, 72, 74, 78, 80–82, 86, 91, 92]. The authors of the corresponding work may be forgiven for wanting to focus on the model part of the detection algorithms by removing threshold choice as a variable, though they fail to state that the results obtained are unobservable in the real-world, as no method of setting a threshold in an unsupervised way is discussed. Sometimes, assumptions are made, such as using a given percentile of the anomaly score from training or validation subsets [47, 52, 58, 79, 83, 85, 94], though the percentile choice is never properly justified. Others [2, 77, 88–90] use the peaks-over-threshold method, an automatic threshold selection procedure based on a generalised Pareto distribution, which still needs to be parameterised and hence requires labelled data. Though rarely,

some publications [69, 84] justify threshold choice by claiming to be set by an expert, but it is not clear how someone, no matter how knowledgeable in their domain, can set a threshold for an uninterpretable and non-dimensional anomaly score without having labelled data. Active learning could serve as an alternative, as a domain expert can actively label windows or sequences which can be used to further improve the detection performance of a model initially trained in a semi-supervised or unsupervised manner. This can be of benefit to the real-world application of the approaches discussed in this work, since querying and acquiring labels with time allows practitioners to gauge the detection performance of the approach used, leading to more transparency in an otherwise unsupervised setting.

Apart from the issues involving the threshold choice, there is also the choice of a suitable window size. The window size is another important hyperparameter, as it defines the amount of temporal information contained in a window. Some work [47, 49, 50, 61, 63, 66, 74, 80, 86, 87] circumvent this by training models on entire sequences, though this can lead to inefficient modelling if no significant dynamic process within the data lasts for the entirety of the time series. Using a window size just long enough to capture all dynamic processes within the data, on the other hand, allows for efficient use of the model's learning capacity. Unfortunately, much of the work published [2, 32, 52, 53, 55–57, 60, 69–73, 75, 77–79, 81–84, 90, 92, 93, 95] does not delve into the process of choosing a window size or use labelled data [48, 58, 59, 66, 76, 85, 89, 94]. Curiously, some publications [75, 79, 82, 89, 94] use a very small window size of less than 10 time steps, raising questions on whether the time series data used has enough temporal correlation to warrant being treated as time series instead of as individual data points. Again, it is not possible to adjust window size outside a supervised setting, and therefore methods, that do not provide a way to find the window size are not applicable in real-world settings. In contrast, only Homayouni et al. [62] use autocorrelation as an unsupervised tool to obtain a plausible window size. For a more in-depth explanation of autocorrelation, refer to Chapter 2.1.

Chapter 4

Datasets

Datasets play a fundamental role in the benchmarking aspect of research. Together with evaluation metrics, they allow for the quantification of a scientific contribution to the field and give readers and practitioners context to how any given approach performs. This chapter introduces the two main datasets used throughout this thesis, along with details on their generation and collection procedure and a preliminary analysis highlighting some key properties.

4.1 Simulation-based Publicly Available Dataset

As thoroughly discussed in Chapter 3.1, the most popular data sets suffer from several issues [31, 33]. While a few datasets were published that have attempted to address the identified issues, they solely represent continuous-sequence problems. Some real-world applications, however, involve discrete sequences. These applications might have patterns in the test data that do not appear in the training data, yet are still considered nominal. Research should aim to provide solutions for real-world problems, though if advances in the research field are disconnected from real-world settings, proposed solutions are of limited value. On the other hand, creating high-quality datasets from real-world problems can be difficult and costly, and hence there is a clear opportunity for generated but real-world inspired datasets to provide a cost-effective way to complete the anomaly detection dataset landscape [33]. This chapter details the generation of a non-trivial, and high-quality discrete-sequence dataset consisting of multivariate time series for online anomaly detection, named the *powertrain anomaly time series benchmark (PATH)* dataset. While primarily aimed at unsupervised anomaly detec-

tion, versions for semi-supervised anomaly detection, and time series generation and forecasting are provided as well. Despite the data being generated using simulation, the underlying electric vehicle simulation model is strongly motivated by the real world and is therefore complex and variable-state. The variable-state nature requires anomaly detectors to make the distinction between behaviour changes due to different states, like the high-voltage battery (HVB) state of charge (SoC) and behaviour changes due to an anomaly, further complicating detection.

4.1.1 Simulation Model

To create an extensive and diverse data set, the use of a physically inspired model from which data can be generated via simulation is proposed. MathWorks offers reference models for a variety of dynamic systems, one of which is the full electric vehicle (FEV) model [96] from the powertrain blockset in MathWorks Simulink. This choice is based on the author's familiarity with the domain, as generating data without any background knowledge may lead to systematic errors. The FEV model offered by MathWorks describes longitudinal dynamics and consists of six main subsystems: the drive cycle block, the driver block, the environment block, the controllers block and the vehicle block. The topology of the FEV model is illustrated in Figure 4.1.

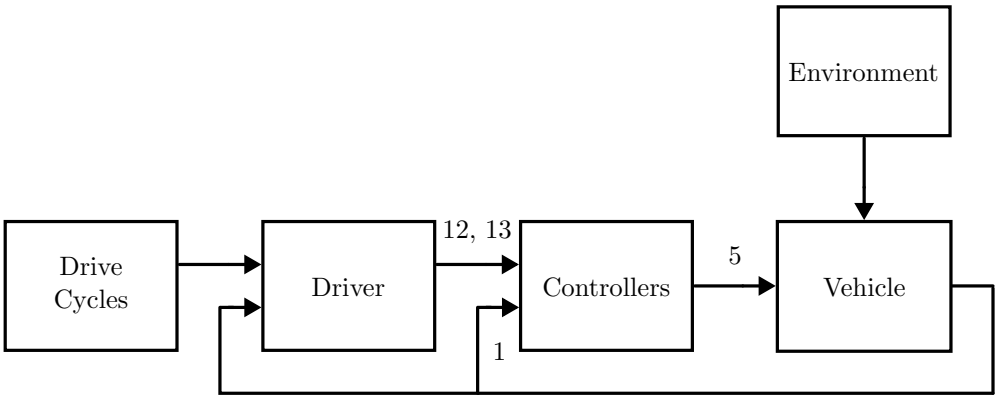

Figure 4.1: Simplified schematic of the FEV model used for the generation of the PATH data set. The numbers represent the indices of signal flow, whose reference is shown in Table 4.1. Only a subset of the signals in Table 4.1 are shown here, as the rest are signals exist exclusively within the vehicle block.

To represent system behaviour, $d_{\mathcal{D}} = 16$ signals are chosen to be logged during simulation and are summarised in Table 4.1. These signals are chosen based on do-

Table 4.1: Signals included in the PATH dataset, along with their physical units and persistent indices. For brevity, electric motor (EM) is abbreviated to EM, henceforth.

Index	Signal Name	Unit	Index	Signal Name	Unit
1	EM Angular Speed	rad/s	9	Axle Force Rear	N
2	EM Torque	N m	10	Axle Speed Front	rad/s
3	Axle Torque Front	N m	11	Axle Speed Rear	rad/s
4	Axle Torque Rear	N m	12	Accelerator Pedal	-
5	HVB SoC	%	13	Brake Pedal	-
6	HVB Current	A	14	HVB Temperature	°C
7	HVB Power	W	15	Cooling Pump Power	W
8	Axle Force Front	N	16	Refrigerator Power	W

main knowledge, with the goal of picking the features that are most representative of powertrain behaviour.

The drive cycle block of the FEV model defines the target vehicle speed and features a series of real-world drive cycles, i.e. profiles depicting the target vehicle speed as a function of time. From the list of speed profiles available in the original FEV model [96], a subset of them is eliminated due to their unrealistic nature, e.g. linearity, as they can be exclusively described using linear functions, or duplicity, as many cycles are present as sub-sequences in others. For example, the FTP72 drive cycle is present within FTP75 and the LA92Short drive cycle is present within LA92. In addition to that, drive cycles aimed at heavy vehicles, like trucks or buses, are also eliminated. The resulting subset of drive cycles chosen for simulation contains 33 different speed profiles of varying length, shown in Table 4.2, along with their lengths in seconds. Some drive cycles may be designed for specific types of powertrains such as diesel ones, but given that they only represent vehicle speed profiles, there is little reason why they cannot be driven by a vehicle with an electric powertrain, like the one modelled in this case.

The driver block of the FEV model regulates the dynamic system to maintain the target speed. Its inputs are the target vehicle speed and the actual vehicle speed, and its outputs are the acceleration and deceleration control commands, index 12 and 13 in Table 4.1, respectively, which are fed into the controllers block of the FEV model. This block takes said accelerator and brake pedal commands stemming as well as vehicle states like actual vehicle speed, EM angular speed, and HVB signals to calculate request signals for the powertrain, like the required EM torque and the brake signal, as well as battery management system signals like the HVB SoC, index

Table 4.2: Drive cycles used for the PATH data set generation, along with their respective lengths in seconds.

Drive Cycle	Length [s]
FTP75	2474
US06	600
SC03	600
HWFET	765
NYCC	598
HUDDS	1060
LA92	1435
IM240	240
UDDS	1369
WLTP Class 1	1022
WLTP Class 2	1477
WLTP Class 3	1800
Artemis Urban	993
Artemis Rural Road	1082
Artemis Motorway 130 kmph	1068
Artemis Motorway 150 kmph	1068
JC08	1204
JC08 Hot	1376
World Harmonized Vehicle Cycle (WHVC)	900
Braunschweig City Driving Cycle	1740
RTS 95	886
ETC FIGE Version 4	1800
CUEDC Petrol cycle	499
CUEDC SPC240 cycle	240
CUEDC diesel cycle - MC	1723
CUEDC diesel cycle - NA	1795
CUEDC diesel cycle - NB	1706
CUEDC diesel cycle - ME	1678
CUEDC diesel cycle - NC	1797
CUEDC diesel cycle - NCH	1676
China Light-Duty Vehicle Test Cycle for Passenger Cars	1800
China Light-Duty Vehicle Test Cycle for Commercial Vehicles	1800
China Worldwide Transient Vehicle Cycle	1799

5 in Table 4.1. Electric vehicles are capable of regenerative braking, meaning, the EM is used to decelerate the vehicle by acting as a generator, thereby charging the HVB if it is not already fully charged.

Following is the vehicle block of the FEV model, which outputs how the vehicle

reacts to any inputs and contains the electric plant subsystem and the drivetrain subsystem. Both take inputs from the controllers block, including the HVB SoC, and the environment block, as well as from each other, as shown in Figure 4.2. The

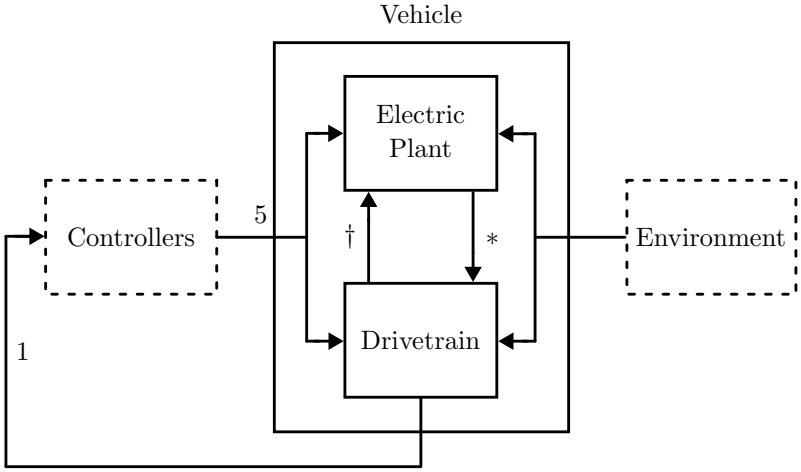

Figure 4.2: A more detailed schematic of the vehicle model depicted in Figure 4.1. Numbers represent the indices of signal flow, reference is shown in Table 4.1. For visual clarity, ∗ is used to represent signals with indices 2, 6, 7, 14, 15 and 16, while † signals with indices 1, 3, 4, 8, 9, 10, 11. Output signals of the vehicle model, which are not fed back into other subsystems, are not shown, for simplicity.

electric plant subsystem outputs the EM torque, the HVB current and power, the HVB temperature and the cooling pump and refrigerator powers, which correspond to indices 2, 6, 7, 14, 15 and 16 in Table 4.1, respectively. The EM torque is input into the drivetrain subsystem, which in turn outputs the EM angular speed, front and rear axle torques, forces and speeds, corresponding to indices 1, 3, 4, 8, 9, 10, 11 in Table 4.1, respectively. The EM angular speed is also fed back into the electric plant model, completing the control loop. Both subsystems also contain further subsystems within them which uncover the causal relationships between their respective signals, but diving as deep as the lowest abstraction level of the model is outside the scope of this thesis. By default, the HVB model features a static temperature model, however, to increase system complexity, a dynamic temperature model is added to the FEV model. The dynamic temperature model used is the electric vehicle HVB cooling system model [97], also from MathWorks.

The environment block of the FEV model dictates environmental conditions that affect the longitudinal dynamics of the FEV model. Parameters like atmospheric pressure, wind speed, road gradient and coefficient of friction can be modified within

this subsystem.

By default, the signals are logged at a sampling frequency of 10 Hz and the solver used is the differential algebraic equations solver for Simscape (daessc). Physical simulations may encounter numerical under- and overflow, which slow down simulations drastically. To avoid simulations getting "stuck", a timeout counter of one hour is set in place to skip the current simulation if triggered. Simulation time depends on the length of the drive cycle, however, for the computer hardware used this is never longer than 20 minutes, and hence one hour is considered sufficient for simulations that do not stall.

Note that, unlike typical internal combustion engine powertrains, this powertrain does not feature discrete gear states, which would decouple the EM angular speed from the axle angular speeds, though a gearbox could be added in future versions of this powertrain for increased complexity. Additionally, further EMs could be added to the model to further increase the complexity of the simulation model.

Readers interested in more detail can refer to the full MathWorks Simulink model available in the GitHub repository under `github.com/lcs-crr/Thesis`. The entire PATH dataset in both its raw and ready-to-use forms can be found in the Zenodo repository under `zenodo.org/records/13255120`.

4.1.2 Data Set Generation

To generate a data set that is not only extensive but also diverse, 100 simulations are undertaken for each of the 33 drive cycles, each with random initial HVB temperatures and HVB SoCs. All simulations are run on a workstation equipped with an Intel Xeon Gold 6234 CPU running Windows 10 Enterprise LTSC version 21H2 with MATLAB 2023b. At this stage, all model properties are left as default, and hence all simulation results are considered *nominal*. For each simulation, the two states (HVB temperatures and HVB SoCs.) are sampled from uniform distributions $\mathcal{U}(10\,^{\circ}\mathrm{C}, 30\,^{\circ}\mathrm{C})$ and $\mathcal{U}(10\,\%, 100\,\%)$, respectively, to ensure no further bias is introduced. This also eliminates and reduces the effectiveness of simple threshold and control chart methods, since the HVB temperature and SoC, but also, by extension and to a lesser extent, other channels, exhibit nominal but high deviation from the average behaviour. As a precautionary measure, drive cycles with a minimum SoC value lower than 5 % are removed, as very low values are observed to result in abnormal behaviour. To add further complexity and to reflect real-world properties, *post-processing* is applied to all time series in this dataset. First, the beginning of time series is trimmed by random

number of time steps so that time series representing the same drive cycle are rarely in sync. The amount by which a given time series is trimmed is sampled from uniform distribution $\mathcal{U}(0, 0.1T)$. This artefact can happen in the real world and means that, for the same drive cycle, any given time step is not comparable across different sequences, eliminating the viability of simple statistical methods such as control charts. In addition to that, noise sampled from Gaussian distribution $\mathcal{N}(0, 0.01\sigma_{\mathcal{D}})$ is added to further move the obtained data towards the real world, where $\sigma_{\mathcal{D}}$ is the feature-wise standard deviation of the data set. After simulation, $N_{\text{n}} = 3273$ highly diverse and unique nominal time series are collected.

For illustration purposes, a nominal time series is plotted in Figure 4.3.

For the generation of *anomalous* time series, six types of anomalies are considered. Some types can occur as both sub-sequence anomalies and whole-sequence anomalies, while others only in whole-sequence anomaly form, due to simulation model limitations. The distribution of sub-sequence and whole-sequence anomalies across the different anomaly types is shown in Table 4.3. Anomalies are caused by changing

Table 4.3: Number of sub-sequence anomalies N_{ss} and number of whole-sequence anomalies N_{ws} by anomaly type.

Anomaly Types	N_{ss}	N_{ws}
Regenerative Braking Off	33	32
Increased Headwind	33	31
Reduced Pump Displacement	1	23
Reduced EM Torque Request	32	33
Increased Wheel Diameter	0	33
Increased Driver Reaction Time	0	33

certain model properties *prior* to simulation, ensuring that any observed anomalous behaviour results from simulation rather than manual tampering of the data, like in the UCR data set [31], which reduces bias. All simulations are done with a fixed seed of 1.

To ensure that the different anomaly types actually lead to anomalous behaviour, control simulations with the same initial HVB states but with otherwise regular model properties are run in parallel. The data from the control simulations are solely used for identification of anomalous behaviour and are naturally excluded from the dataset.

For the first kind of anomaly, the regenerative braking is turned off, which leads to visibly different EM and axle torques, as well as HVB current and power, as these can no longer assume negative values. When regenerative braking is off, the HVB SoC

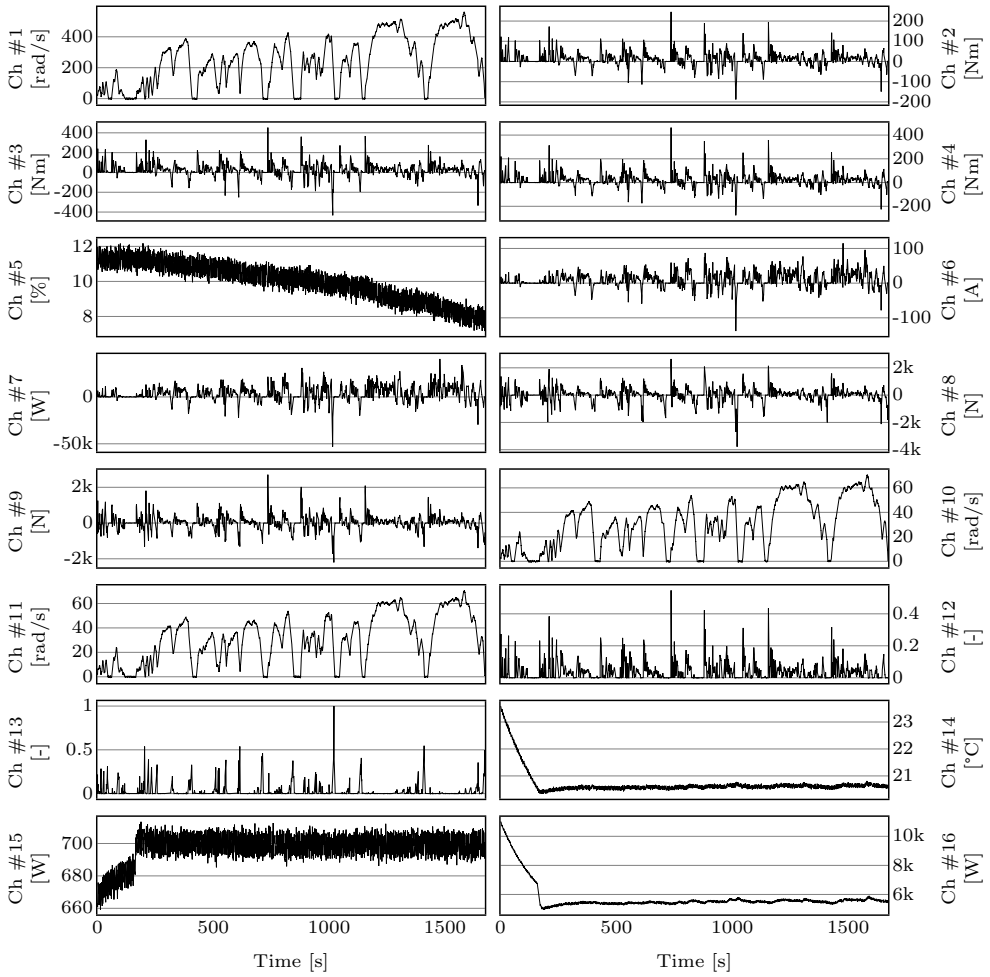

Figure 4.3: Sample plot of an anomaly free sequence with added noise and undergone trimming. The channel legend can be found in Table 4.1.

now has an exclusively negative gradient as it is no longer recharged via regenerative braking, and hence it decreases at a faster rate. The brake pedal is also used more to compensate for the missing EM torque for braking. For each of the cycles, this anomaly type is simulated in two different ways: without regenerative braking from the beginning and from a random point in time within the drive cycle. This random point in time is sampled from a uniform distribution $\mathcal{U}(0.2T, 0.8T)$, where T denotes the temporal length of the respective drive cycle, see Table 4.2. This distribution is used for all other sub-sequence anomaly types. Clearly, this introduces a positional

bias, which always leads to a transition from nominal to anomalous behaviour. While this bias can easily be removed, it is kept, as this type of dynamic system would also most likely feature this bias in the real world. As discussed in Chapter 3.1, there are no metrics that are aware of detection delay and compatible with discrete-sequence problems. A set of metrics addressing the two short-comings are proposed, though they are also not perfect and can only be applied to problems with this positional bias. One of the anomalous time series for the CADC130 drive cycle and its control simulation counterpart are plotted in Figure 4.4.

In the case of the next anomaly type, a headwind of 5 m/s is simulated. This headwind acts as a resistive force on the frontal area of the vehicle and needs to be overcome to maintain the target vehicle speed. The driver model approaches this by using the accelerator pedal more than the norm, which leads to higher EM and axle torques and therefore axle forces. The higher EM torque requires a higher HVB current and power, which also causes accelerated discharging. Like previously, this anomaly type is simulated for each drive cycle, both from the beginning and from a random point in time within the cycle. One of the anomalous time series for the CLTCPassenger drive cycle and its control simulation counterpart are plotted in Figure 4.5.

Following that, the displacement of the cooling pump is reduced by 10 % to simulate another anomaly type. Evidently, this leads to a higher HVB temperature as the cooling capacity is reduced. This reduction is also visible in the pump power. Like with the previous two anomaly types, this anomaly type can start from the beginning and from a random point in time within the cycle. One of the anomalous time series for the CUEDCDieselME drive cycle and its control simulation counterpart are plotted in Figure 4.6.

For the next anomaly type, the requested EM torque value output by the powertrain control module is also reduced by 10 %. As a response to this, the driver model requests a higher acceleration pedal value and consequently a different brake pedal values as well. This anomaly type can also start from the beginning and from a random point in time within the cycle. One of the anomalous time series for the FTP75 drive cycle and its control simulation counterpart are plotted in Figure 4.7.

In the next case, the wheel diameter under weight is increased by 10 % which, for the same target vehicle speed, leads to a lower EM and axle angular speed. Furthermore, a larger wheel diameter leads to higher EM and axle torques, which are achieved using higher accelerator and brake pedal values. Here, the wheel diameter also has an effect on the HVB temperature, which, depending on its absolute magnitude, may also affect the cooling system. Due to model limitations, this anomaly can only be

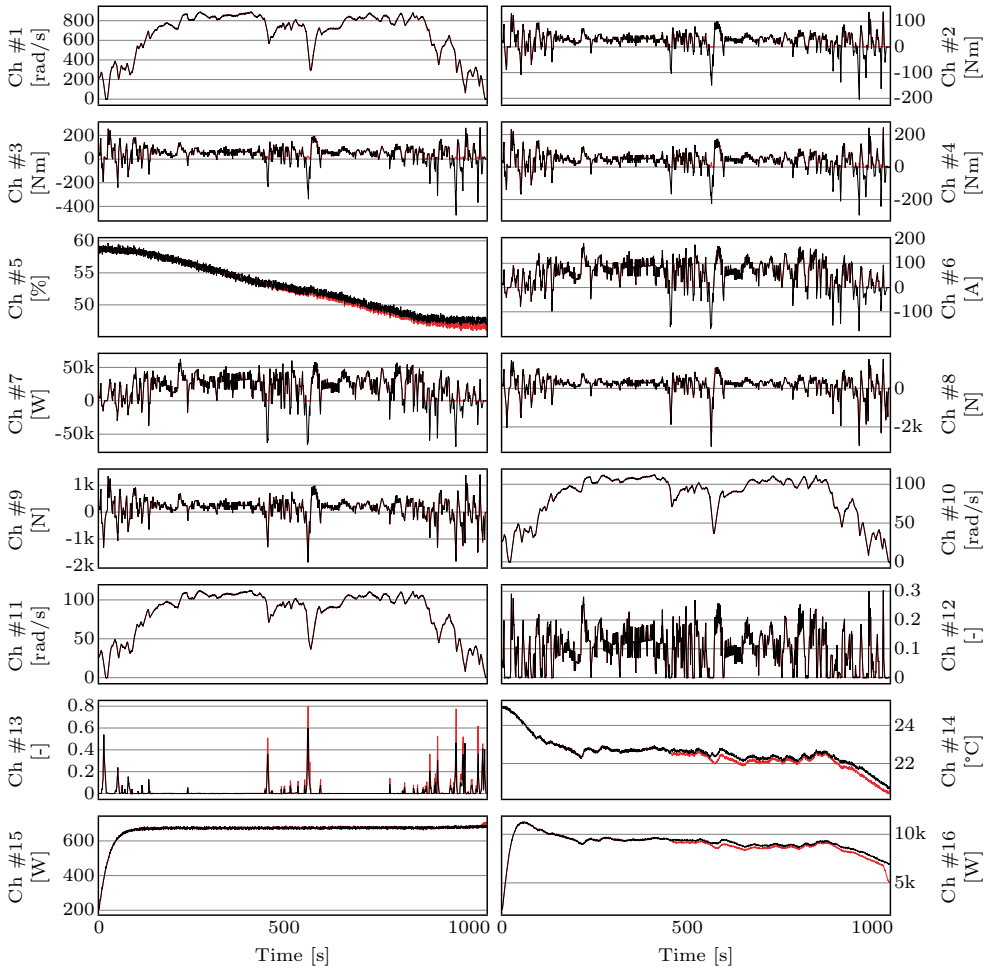

Figure 4.4: Plot of an anomalous sequence without regenerative braking (in red) and its control simulation counterpart (in black), both with added noise and undergone trimming. The anomalous sub-sequence starts after 384.6 s. The channel legend can be found in Table 4.1.

simulated for whole-sequence anomalies. One of the anomalous time series for the HUDDS drive cycle and its control simulation counterpart are plotted in Figure 4.8.

The last anomaly is recorded by increasing the driver response time by a factor of 4. This is one of the more subtle anomalies types, but manifests itself in all channels, except for the HVB temperature and cooling. Like for the wheel diameter anomaly, this anomaly can only be simulated for whole-sequence anomalies. One of the anomalous time series for the LA92 drive cycle and its control simulation counterpart are plotted

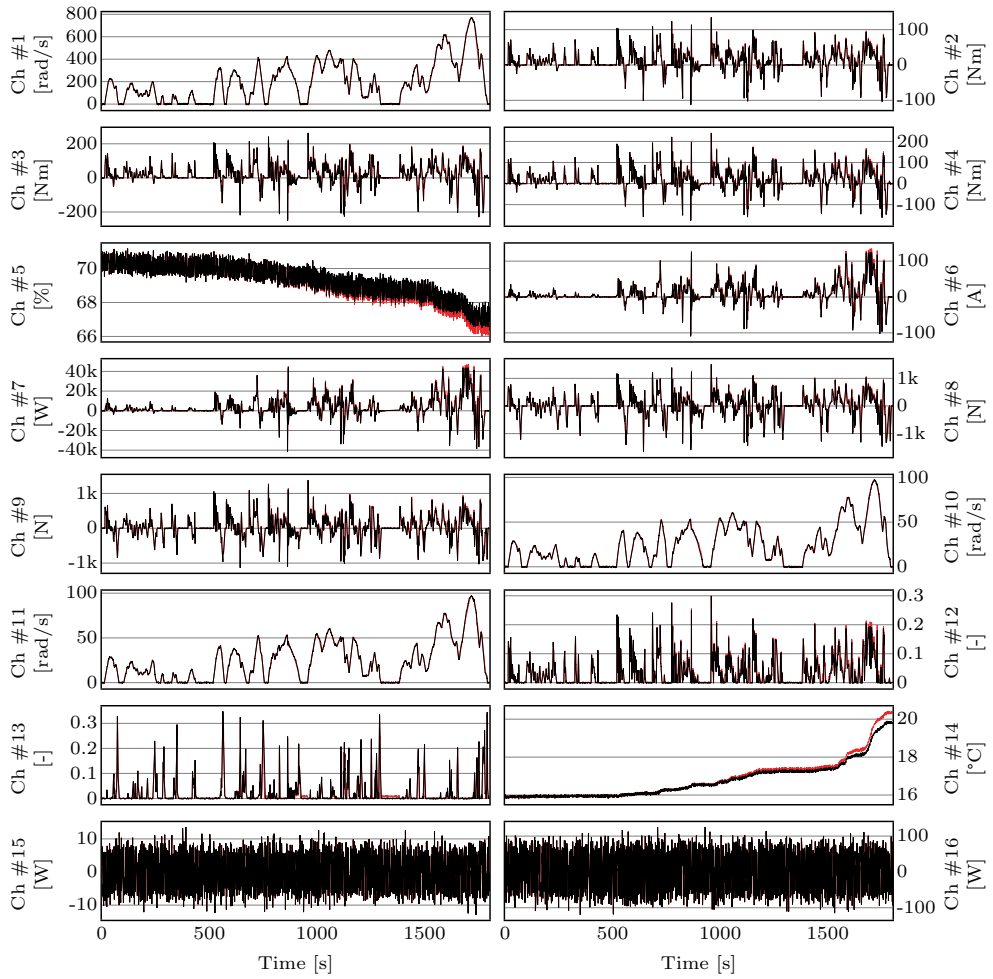

Figure 4.5: Plot of an anomalous sequence with an added headwind (in red) and its control simulation counterpart (in black), both with added noise and undergone trimming. The anomalous sub-sequence starts after 738.0 s. The channel legend can be found in Table 4.1.

in Figure 4.9.

Recall the findings by Wenig et al. [34] in Chapter 3.1. As outlined above, all anomaly types manifest themselves in more than one channel, making the generated dataset truly multivariate with multivariate anomalies. This further underlines the real-world-motivated nature of the dataset, since properties of complex real-world systems also correlated with one another.

Given the uniform distribution from which the HVB temperature is sampled from, half of the simulated anomaly types start with a HVB temperature below 20 °C. In

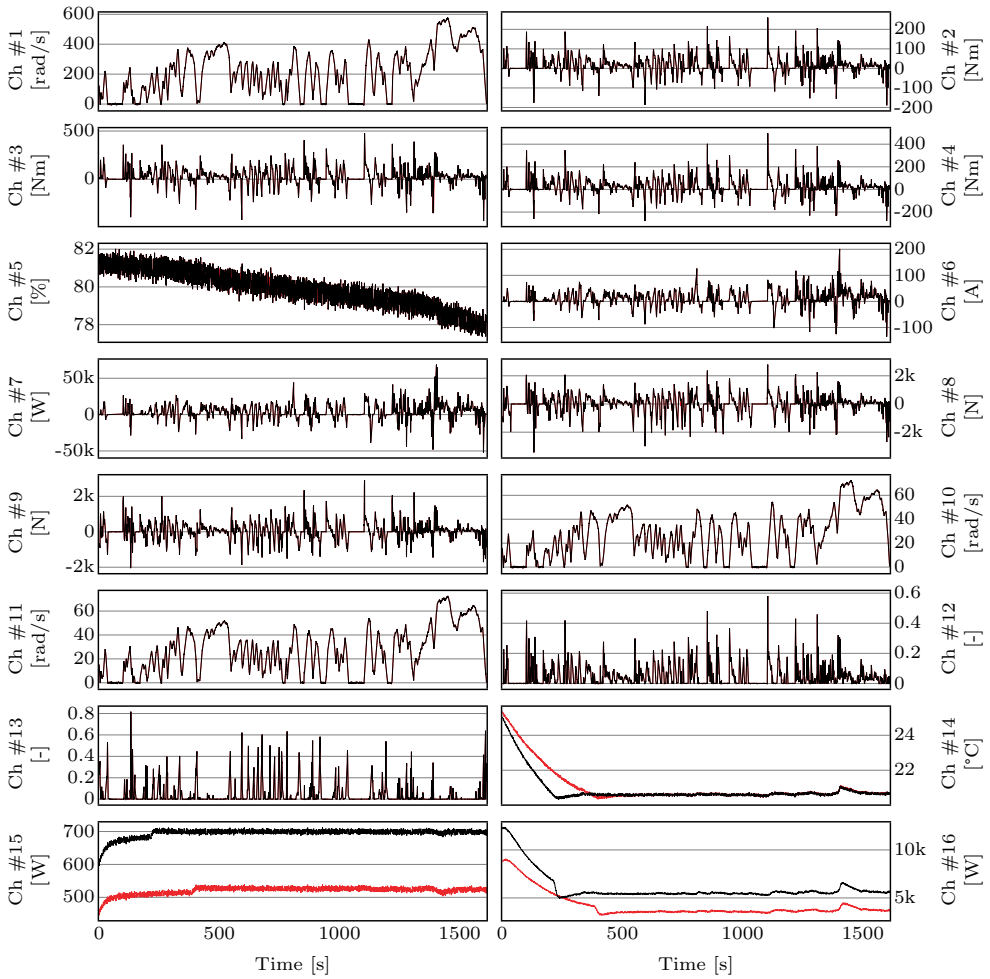

Figure 4.6: Plot of an anomalous sequence with a reduced cooling pump displacement (in red) and its control simulation counterpart (in black), both with added noise and undergone trimming. It is a whole-sequence anomaly, and hence the anomalous behaviour starts from the first time step. The channel legend can be found in Table 4.1.

these cases, the HVB will naturally heat up as it is being used, and hence the cooling system does not play a role in short drive cycles. Therefore, in the case of the reduced cooling pump displacement, often no anomalous behaviour can be observed because the simulated anomaly is identical with the corresponding control simulation. For these cases, the simulated anomaly is discarded.

Finally, this results in $N_{\mathrm{a}} = 284$ successful simulations yielding anomalous time series, where $N_{\mathrm{a}} = N_{\mathrm{ss}} + N_{\mathrm{ws}}$, N_{ss} denotes the number of sub-sequence anomalies and

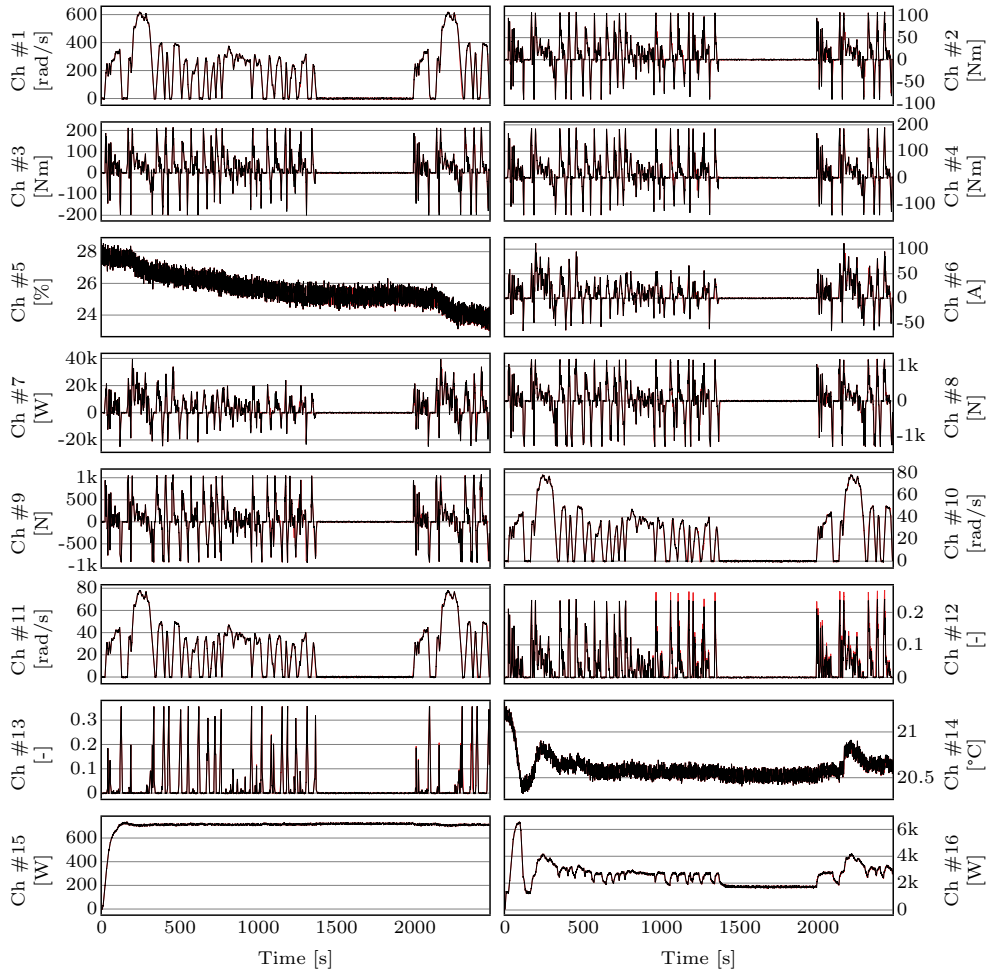

Figure 4.7: Plot of an anomalous sequence with a reduced requested EM torque (in red) and its control simulation counterpart (in black), both with added noise and undergone trimming. The anomalous sub-sequence starts after 940.2 s. The channel legend can be found in Table 4.1.

N_{ws} the number of whole-sequence anomalies. Hence, the entire data set $\mathcal{D}$ consists of $N_{\mathrm{n}} + N_{\mathrm{ss}} + N_{\mathrm{ws}} = 3557$ unique (nominal and anomalous) multivariate time series, with an anomalous sequence ratio of $N_{\mathrm{a}}/N \approx 8\,\%$. This amount of anomalous data in relation to anomaly-free data is used as it approximately matches the anomaly ratio in public data sets [2–4, 32] and fulfils the assumption that anomalous behaviour is rare by nature [33]. $\mathcal{D}$ is then shuffled and divided into three separate folds for cross-validation, which corresponds to a 67/33 split for training and test, respectively.

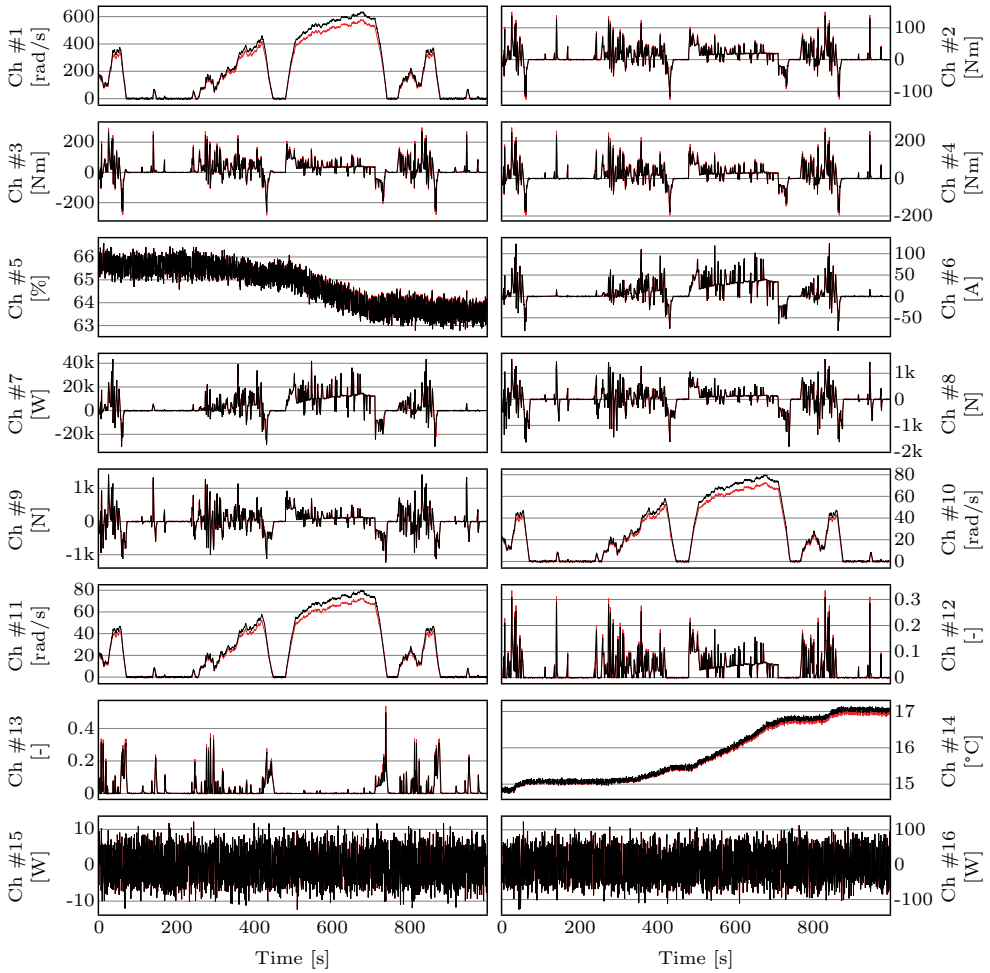

Figure 4.8: Plot of an anomalous sequence with an increased loaded wheel diameter (in red) and its control simulation counterpart (in black), both with added noise and undergone trimming. It is a whole-sequence anomaly, and hence the anomalous behaviour starts from the first time step. The channel legend can be found in Table 4.1.

Formally, the training subset $\mathcal{D}^{\text{train}} = \{\mathcal{S}_1, ..., \mathcal{S}_m, ..., \mathcal{S}_M\}$ then consists of $M = 2371$ multivariate time series on average, where $\mathcal{S}_m \in \mathbb{R}^{T_m \times d_{\mathcal{D}}}$, where T_m is the number of time steps in sequence $\mathcal{S}_m$. Likewise, the test subset $\mathcal{D}^{\text{test}} = \{\mathcal{S}_1, ..., \mathcal{S}_n, ..., \mathcal{S}_N\}$ consists of $N = 1186$ multivariate time series on average, where $\mathcal{S}_n \in \mathbb{R}^{T_n \times d_{\mathcal{D}}}$, where T_n is the number of time steps in sequence $\mathcal{S}_n$. For benchmarking purposes, the use of the prescribed training and test split are strongly recommended.

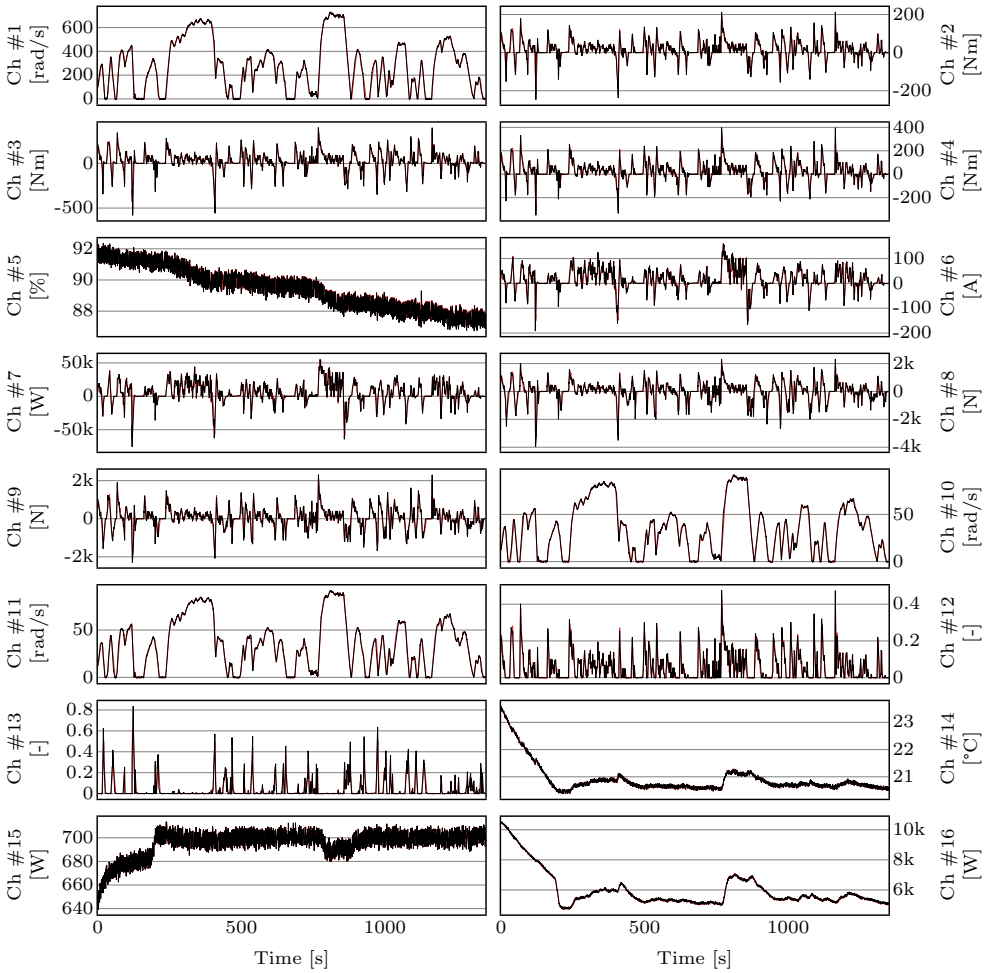

Figure 4.9: Plot of an anomalous sequence with increased an driver response time (in red) and its control simulation counterpart (in black), both with added noise and undergone trimming. It is a whole-sequence anomaly, and hence the anomalous behaviour starts from the first time step. The channel legend can be found in Table 4.1.

4.1.3 Usage of the Data Set

Clearly, both $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$, as specified above, contain nominal and anomalous time series, though in an unsupervised setting the labels for $\mathcal{D}^{\text{train}}$ should be disregarded. This is because, this data set is aimed at *unsupervised* time series anomaly detection, which requires approaches especially robust to contaminated training data.

Apart from the unsupervised setting, the underlying properties of the data set can

be useful in other research areas. The same $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$ subsets can also be used for *imbalanced time series classification*. For *semi-supervised* anomaly detection, a clean $\mathcal{D}^{\text{train}}$ with on average $M = 2182$ nominal time series and the same labelled $\mathcal{D}^{\text{test}}$ is provided in the data set, where clean refers to the absence of anomalous sequences in the subset. Furthermore, for time series *forecasting* or *generation*, clean versions of both $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$ are provided, where $M = 2182$ and $N = 1091$ on average, respectively. Despite being targeted at *online* time series anomaly detection, the PATH data set can just as well be used in *offline* time series anomaly detection. For an explanation of online and offline anomaly detection, please refer to Chapter 2.2.

4.1.4 Preliminary Analysis

To provide a more complete picture to the reader, some preliminary analysis on the generated dataset is provided. As mentioned above, the simulation model has a sampling frequency 10 Hz, meaning that time steps of the resulting time series are 0.1 s apart. According to the Nyquist–Shannon theorem [98], a signal sampled at an arbitrary sampling frequency can only represent dynamic process frequencies of at most half the sampling frequency, which, in this dataset, is 5 Hz.

Fourier transformations allow for the decomposition of time series data into its components in the frequency domain. For a brief explanation of the Fourier transformation, please refer to Chapter 2.1. To find which frequencies contain the most information, a Fourier transformation is performed for each feature and for each driving cycle. Summing the amplitudes of all features for each driving cycle reveals that, most information is represented in the lower frequency range, as exemplified in Figure 4.10. Another observation is how complex the drive cycles are. Consider the simple combination of sine waves at 1 Hz and 2 Hz, a cosine wave at 5 Hz, and some minor noise at 20 Hz, as shown in the top half of Figure 4.11. Its Fourier transform, shown in the bottom half of Figure 4.11, clearly uncovers the underlying frequencies within the data; see the three peaks at the individual component frequencies as well as the minor peak caused by noise at 20 Hz. In contrast, Figure 4.10 underlines how much more complex the underlying time series is, as illustrated by all the different frequencies that make up the signal.

Since most information is present in the lower frequencies of the data, computational effort can be reduced through downsampling. To avoid aliasing and maintain signal integrity, downsampling is performed by low-pass filtering the time series data. The most important parameter of a low-pass filter is the cut-off frequency, above which

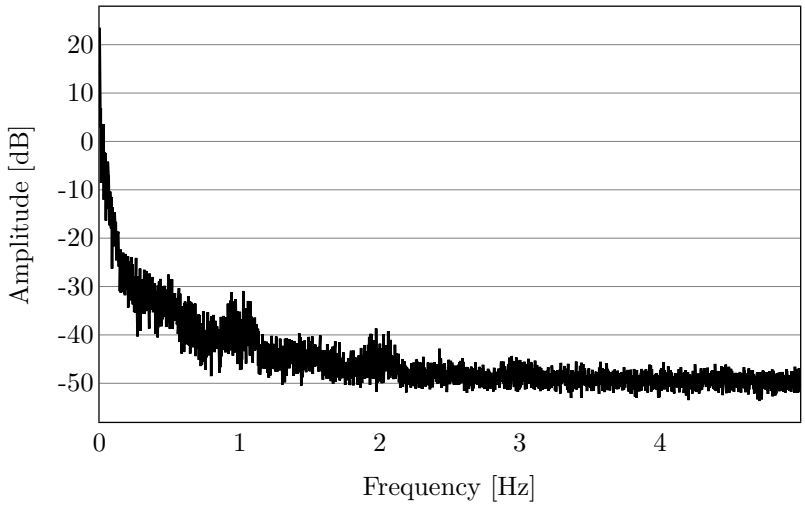

Figure 4.10: Fourier plot of the WLTP1 drive cycle from the PATH dataset. The amplitude seen is the sum of the amplitudes for each feature.

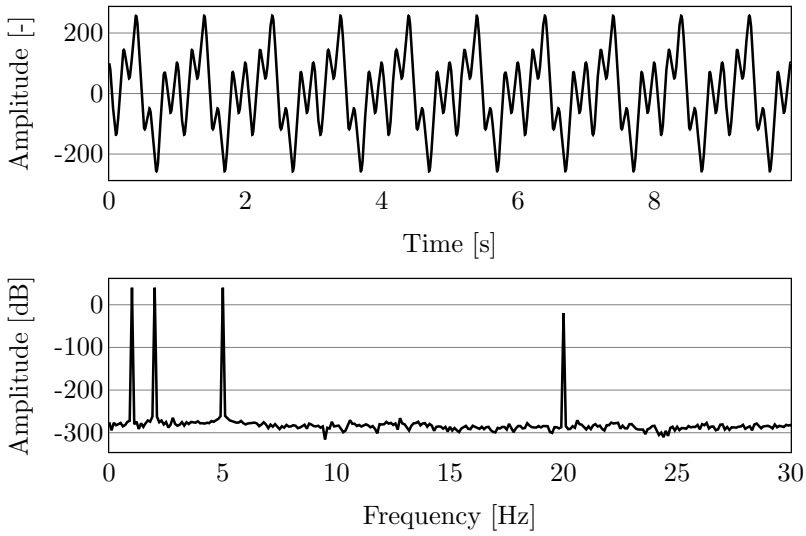

Figure 4.11: Fourier plot of a simple combination of a sine waves.

all frequencies attenuated and, in case of an ideal low-pass filter, removed. The cut-off frequency should be set to at most half of the sampling rate to be downsampled to, as to comply with the Nyquist–Shannon theorem [98]. To quantify the impact of downsampling on the data, various cut-off frequencies are iterated through, and the mean-squared error (MSE) between the original and the filtered time series is

computed. The goal is to find a cut-off frequency which strikes the balance between computational effort, but also fidelity compared to the original data. Figure 4.12 shows the average MSE across all sequences in all folds. As expected, the MSE is

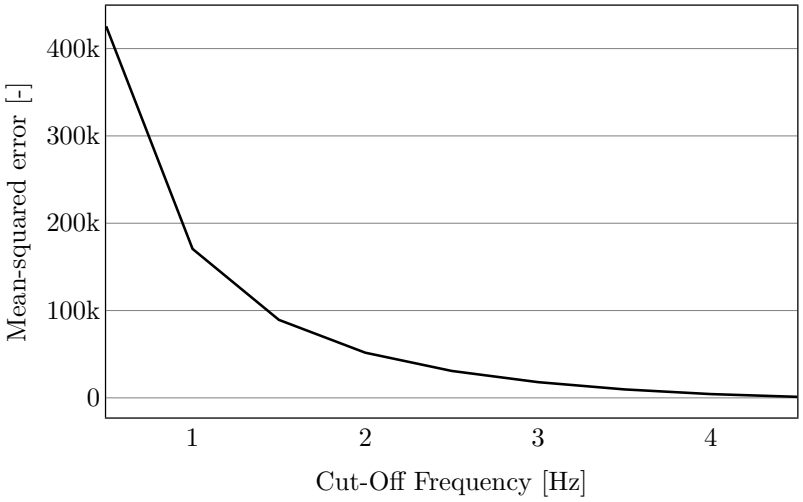

Figure 4.12: Mean-squared error as a function of cut-frequency used in the low-pass filter.

highest at very low cut-off frequencies and decreases rapidly as the cut-off frequency increases. Parameterising the low-pass filter with a cut-off frequency below 1 Hz is therefore undesired, and hence downsampling the dataset to 2 Hz is recommended.

Once the time series data is downsampled, the variable-length sequences can be windowed to create a set of fixed-length sub-sequences, also known as *windows*. The reasoning behind using windows rather than entire sequences is that most of the dynamic processes in the time series data are fairly fast and hence only have a short-term effect that lasts for a small period of time. Modelling entire variable-length sequences would be possible, but much of the model learning capacity would be wasted on internal state mechanisms carrying information for longer periods of time than necessary. This therefore allows for more efficient model training on windows that are only as long as they need to be to represent all present dynamics in the time series data. To find the effect length of the slowest dynamics, an autocorrelation analysis is undertaken for each of the drive cycles and each of the features within them. The pseudocode representing the windowing procedure is shown in Algorithm 1 and is also denoted as the `window` function.

This contrasts with approaches taken in the literature, which treat window size as a hyperparameter that is tuned. An autocorrelation analysis yields the correlation

Algorithm 1 Windowing Process (`window`)

Require: Sequence $\mathcal{S}_n \in \mathbb{R}^{T_n \times d_\mathcal{D}}$, Window Size $w \in \mathbb{N}$, Window Shift $k \in \mathbb{N}$

1: $\lambda \leftarrow (T_n - w)/k + 1$ ▷ Find the total number of windows in $\mathcal{S}_n$
2: **for** $i = 1 \rightarrow \lambda$ **do** ▷ Iterate through total number of windows
3: $\quad \mathbf{X}[\lambda] \leftarrow \mathcal{S}_n[i * k : i * k + w]$ ▷ Assign current window
4: **end for**
5: **return** Window Matrix $\mathbf{X} \in \mathbb{R}^{\lambda \times w \times d_\mathcal{D}}$

between the time series and a range of lagged versions of the time series for a specified number of lags and does not require labels. For a more detailed explanation of the autocorrelation analysis, please refer to Chapter 2.1. The autocorrelation plot for the slowest signal of all sequences in the dataset is shown in Figure 4.13.

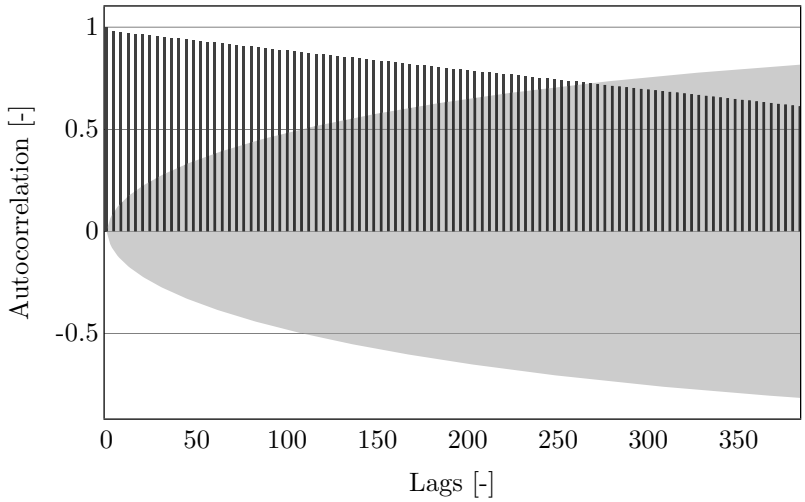

Figure 4.13: Autocorrelation of the SoC signal for the FTP75 as a function of lags. The band shown represents the confidence interval, below which the autocorrelation is no longer statistically significant.

As is evident, the autocorrelation decreases with the number of lags, which is consistent with theory as dynamic effects dampen with time. Therefore, the slowest dynamics present that is still significant is at the lag where the autocorrelation intersects the confidence interval. It is no surprise that the SoC signal features the slowest dynamics; recall Figure 4.3, where the slow dynamics is especially visible. Therefore, the window size parameter should be set as the temporally furthest-away but statistically significant lag of the signal with the slowest dynamics in all sequences, which for the PATH dataset is 268 time steps. Given that some approaches in the benchmark-

ing section strictly require windows to be a multiple of 2, the window size is therefore rounded to $w = 256$, representing 128 s at a 2 Hz sampling rate. Of course, the decision on a suitable sampling rate could also be done manually by a domain expert, who is interested in anomalous behaviour at specific time scales.

4.2 Real-world Application: Endurance Testing

To complement the results obtained with the generated PATH dataset, a dataset from the real world is also experimented with. The dataset stems from the endurance testing of an automotive battery-electric powertrain, a procedure which aims to simulate real-world use by the customer over several years by running tests with little downtime for a prolonged period of time, thereby accelerating powertrain ageing in a shorter period of time. Given the compressed nature of this type of testing, large amounts of data can be collected. Current monitoring systems are rule-based, which can only detect anomalies outside the operating range, do not take signal dynamics into account and represent a high level of manual parameterisation. Therefore, such a real-world scenario is especially suited for the application of a time series anomaly detection algorithm.

4.2.1 Automotive Powertrain Testing Setup

During endurance testing, a set of exclusively proprietary drive cycles is run, which differs from the public drive cycles used in the PATH dataset in Chapter 4.1. The reason proprietary drive cycles are used for endurance runs is that they allow for more extensive loading of the powertrain than public drive cycles, which are mainly used for fuel or energy consumption certification. The portfolio consists of eight dynamic drive cycles representing short, long, fast, slow and dynamic trips ranging from 5 to 30 minutes. Like the PATH dataset, this dataset also contains variable-state behaviour caused by the presence of a HVB in the system.

Aside from the HVB, the testee consists of other components too. The inverter converts the direct current from the HVB into alternating current, which leads to the EM. The EM is then connected via drive shafts to resistance machines which simulate the road load. The EM, inverter, and powertrain controller are cooled via the same loop, and hence their temperatures have some correlation, while the HVB is cooled using a separate loop. A simplified schematic of the components and their relationship is shown in Figure 4.14.

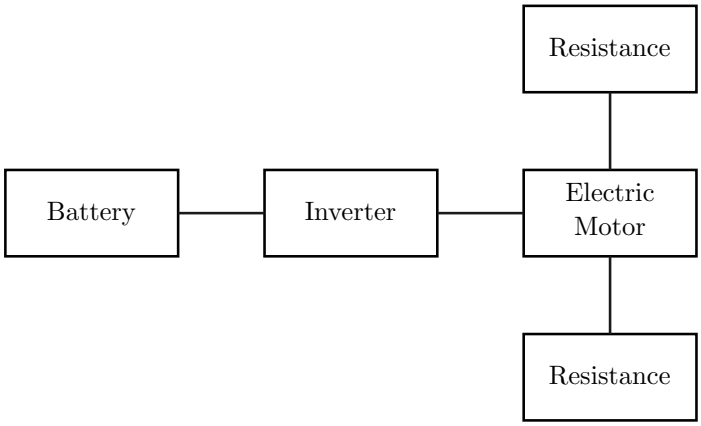

Figure 4.14: High-level schematic of powertrain. Direct current flows from the HVB to the inverter, which in turn converts it to alternating current for the EM. The EM drives two axles, each connected to a resistance.

4.2.2 Data Set

The nominal subset of this real-world data set consists of 1875 of variable-length sequences, which are divided into three folds to enable cross-validation, where 2/3 are used for training and 1/3 for testing. Therefore, the unlabelled training subset $\mathcal{D}^{\text{train}}$ in each fold consists of $M = 1250$ unlabelled time series, where $\mathcal{D}^{\text{train}} = \{\mathcal{S}_1, ..., \mathcal{S}_m, ..., \mathcal{S}_M\}$. For this work, a list of representative $d_{\mathcal{D}} = 22$ channels is hand-picked in consultation with test bench engineers. The chosen features along with their indices are as shown in Table 4.4, while a plot of one anomaly-free measurement is shown in Figure 4.15.

The testing subset $\mathcal{D}^{\text{test}}$ consists of N labelled time series such that $\mathcal{D}^{\text{test}} = \{\mathcal{S}_1, ..., \mathcal{S}_n, ..., \mathcal{S}_N\}$. Recall that the nominal portion of the testing subset $\mathcal{D}^{\text{test}}$ accounts for $N_{\text{n}} = 625$ measurements. Due to the absence of labelled anomalies in the dataset, realistic anomalous events are intentionally provoked and recorded following the advice of test bench engineers. To this end, four anomaly types are recorded. Every anomaly type is recorded for every drive cycle at least once, leading to $N_{\text{a}} = 47$ anomalous measurements that are all used as the labelled anomalous portion of the testing subset $\mathcal{D}^{\text{test}}$. Hence, the testing subset $\mathcal{D}^{\text{test}}$ in each fold is made up of $N = N_{\text{n}} + N_{\text{a}} = 672$ measurements on average, representing an anomaly sequence ratio of around $N_{\text{a}}/N \approx 7\,\%$, putting it in line with the PATH dataset and other public datasets [2–4, 32].

In the first type, the virtual wheel diameter is changed, such that the resulting

Table 4.4: Channels (features) chosen for modelling testee behaviour. The indices correspond to Figure 4.15.

Index	Signal Name	Index	Signal Name
1	Vehicle Speed	12	Inverter Inlet Temperature
2	HVB Voltage	13	Inverter Outlet Temperature
3	HVB Current	14	Powertrain Controller Temperature
4	HVB Temperature	15	Left Axle Torque
5	HVB SoC	16	Right Axle Torque
6	EM Voltage	17	Left Axle Angular Speed
7	EM Current	18	Right Axle Angular Speed
8	EM Torque	19	Cooling Loop 1 Inlet Temperature
9	EM Angular Speed	20	Cooling Loop 1 Outlet Temperature
10	EM Stator Temperature	21	Cooling Loop 2 Inlet Temperature
11	Inverter Temperature	22	Cooling Loop 2 Outlet Temperature

vehicle speed deviates from the norm. The wheel diameter is an adjustable parameter, rather than a physical quantity, as resistances are connected to the shafts rather than actual wheels. This accounts for 16 whole-sequence anomalies and the only channel that demonstrates anomalous behaviour is the vehicle speed, since:

$$v_{\text{vehicle}} = r \cdot \omega \tag{4.1}$$

Here, r is the wheel radius and ω the angular speed of the drive shaft. Logically, the anomalous behaviour is most visible at higher speeds.

In the case of the next anomaly type, the recuperation level is turned from maximum to zero, and hence the minimum EM, and therefore left and right axle torques are always non-negative and the HVB SoC experiences a higher drop. This anomaly type accounts for another eight whole-sequence anomalies.

For the following anomaly, the HVB is swapped for a battery simulator, where the voltage behaviour deviates from a real HVB. Considering that operation works by requesting a given amount of power, a different voltage behaviour will also result in a change in current, since power P is the product of voltage across the HVB V and the current I:

$$P = V \cdot I \tag{4.2}$$

Therefore, this type of anomaly will be evident in the HVB and EM voltage channels, as well as the respective current channels, representing 15 whole-sequence anomalies.

The inverter and EM share a cooling loop, whose cooling capacity is reduced at the

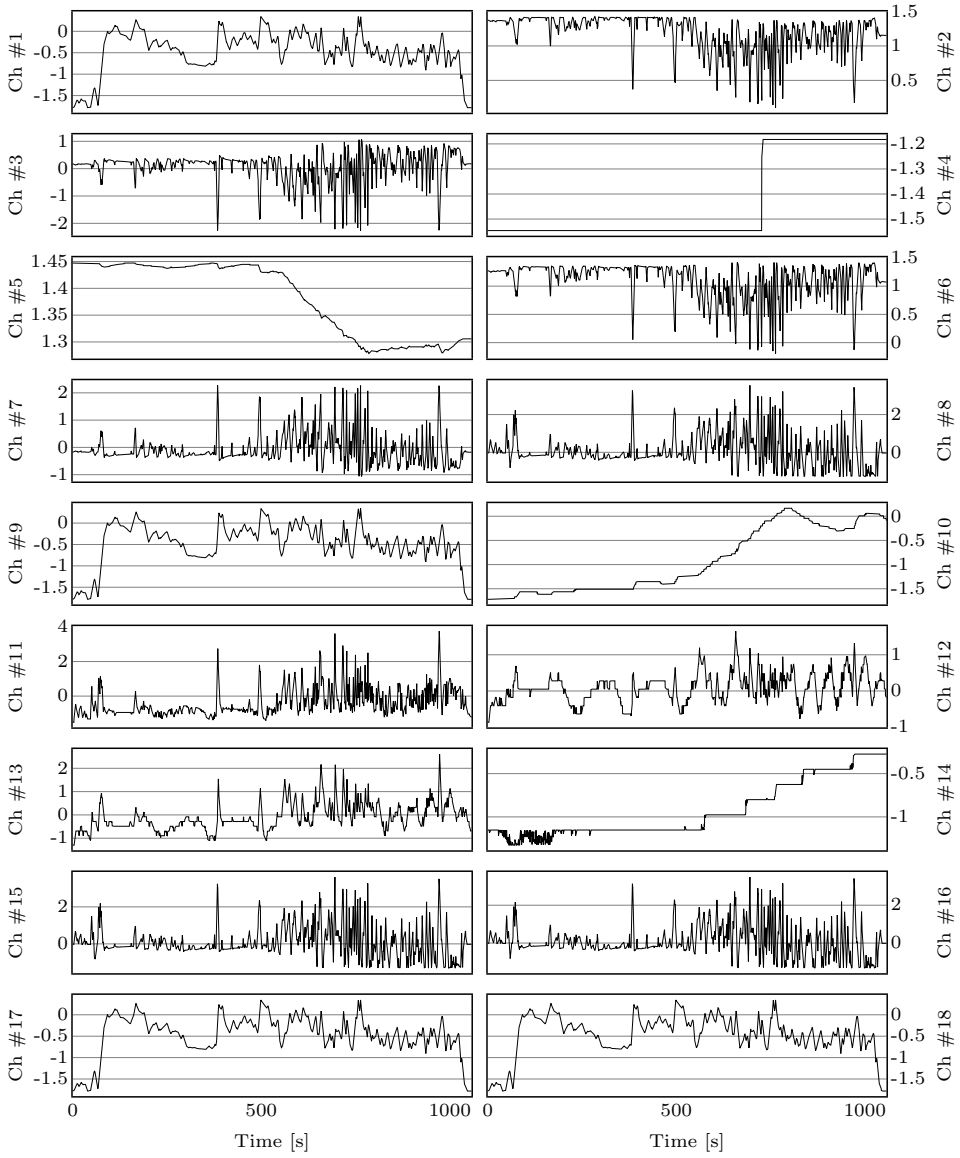

Figure 4.15: Sample plot of an anomaly free sequence. The channel legend can be found in Table 4.4. Note that, the cooling loop inlet and outlet temperatures are omitted due to space constraints. Unlike Figure 4.3, the y-axis ticks are removed due to comply with confidentiality requirements.

beginning or middle of the measurement for the last type of anomaly. This leads to, for instance, higher EM rotor, EM stator, and inverter temperatures than normal. For

three of the sequences, the capacity is reduced at the midpoint of the measurement, whereas another five sequences are recorded with reduced cooling capacity from the very beginning.

In an operative environment, it is desirable to find out whether the previously recorded sequence had any problems to trigger an inspection before the next measurement is recorded. Furthermore, a model that performs as well as possible with as little data as possible translates to faster deployment. Good performance is indicated by a model that can detect as many anomalies as possible and rarely labels anomaly-free measurements wrongly.

4.2.3 Preliminary Analysis

Analogous to Chapter 4.1, a preliminary analysis is also undertaken on the real-world dataset.

Performing a Fourier transformation for each feature and summing their amplitudes reveals that the same observations are made as with the PATH dataset. For the real-world dataset, most information is also represented in the lower frequency range, as shown in Figure 4.16.

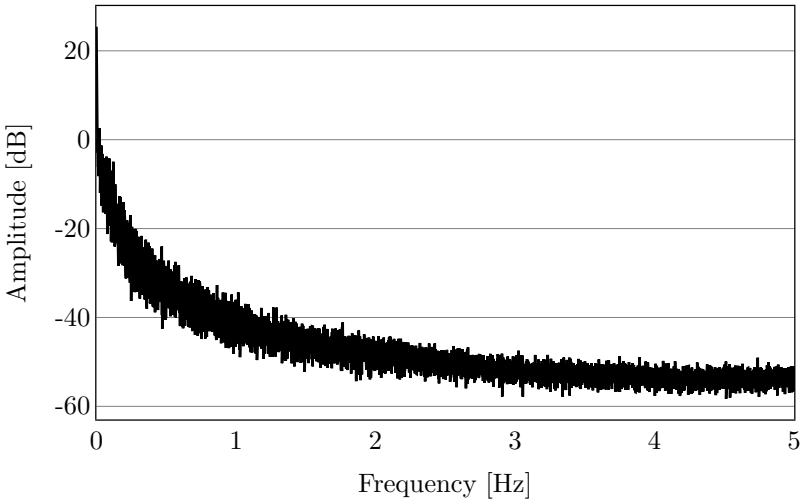

Figure 4.16: Fourier plot of an example drive cycle from the real-world dataset. The amplitude seen is the sum of the amplitudes for each feature.

The fact that the same observations are made in both a real-world dataset and a generated one further supports the claim of the PATH dataset being real-world motivated despite being synthetic.

Like in Chapter 4.1, the impact of downsampling on the data is quantified by iterating through various cut-off frequencies and computing the MSE between the original and the filtered time series. Figure 4.17 shows the average MSE across all sequences in all folds.

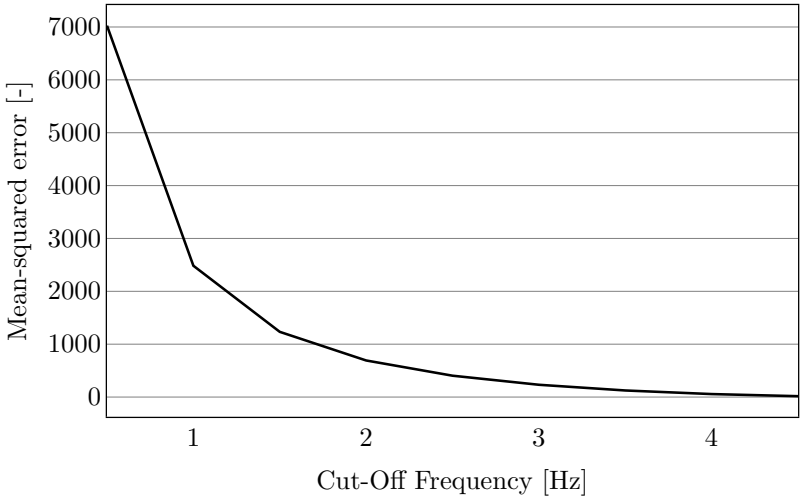

Figure 4.17: Mean-squared error as a function of cut-frequency used in the low-pass filter.

Similarly, the MSE is also highest at very low cut-off frequencies, and hence downsampling the dataset to 2 Hz is recommended as well.

Lastly, a suitable window size is also computed using an autocorrelation analysis. Again, the autocorrelation plot of the slowest channel of all sequences in the dataset is shown in Figure 4.18.

Unlike the PATH dataset, the cooling loop 1 inlet temperature has even slower dynamics than the SoC. Its autocorrelation is statistically significant up to 304 lags, and hence the window size is, again, rounded to $w = 256$ time steps.

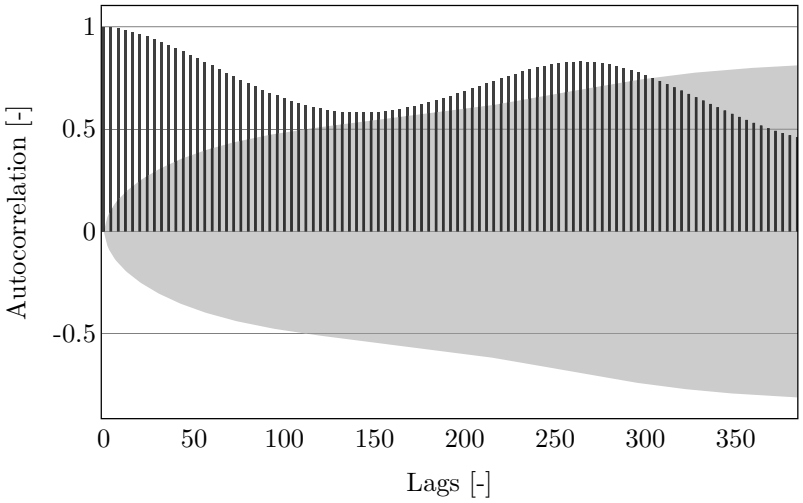

Figure 4.18: Autocorrelation of the SoC signal for drive cycle B as a function of lags. The band shown represents the confidence interval, below which the autocorrelation is no longer statistically significant.

Chapter 5

Online Multivariate Time Series Anomaly Detection

5.1 Introduction

In this chapter, a novel data-driven approach for anomaly detection in time series is proposed. Fundamentally, it consists of the temporal VAE (TeVAE), which builds on the variational autoencoder architecture introduced in Chapter 2.4 and relies on bidirectional long short-term memory (BiLSTM) layers, presented in Chapter 2.3, to model dynamic properties within time series data. TeVAE is further enhanced by multi-head attention, a mechanism which weighs correlated time steps more, in theory facilitating modelling of longer sequences. In the following section, Chapter 5.2, TeVAE's characteristics, design choices, as well as associated experiments are introduced. In literature, architectures featuring multi-head attention are observed to suffer under the so-called *bypass phenomenon* [99], and hence this problem is introduced, along with its relevance for TeVAE. Another component of the approach is a novel method to remap fixed-length windows to variable-length sequences, so-called reverse-windowing. It is especially useful for evaluation as, compared to evaluating the time series windows output by TeVAE, reverse-windowing allows for a higher degree of interpretability for domain experts. Furthermore, it is also outlined how the novel reverse-window method (RWM) is compared with other, more conventional methods. Additionally, it is of interest how well TeVAE can distinguish unseen, but nominal patterns from anomalous ones. The reasoning behind such an experiment and the setting itself are

also discussed. Following that, more generic experimental settings are outlined, like the issue of unsupervised threshold estimation, online detection performance quantification and testing environment. Moreover, Chapter 5.3 presents the results for the real-world-inspired dataset generated using simulation, as well as for the real-world dataset. Here, TeVAE's detection performance is held up against a range of state-of-the-art anomaly detection approaches, whose source code was either freely available, or whose respective publication allows for re-implementation. Finally, the findings obtained are summarised and conclusions are drawn in Chapter 5.4.

Mathematically, time series anomaly detection in discrete-sequence settings can be defined as follows. Recall dataset $\mathcal{D}$ containing data from an arbitrary discrete-sequence problem. It consists of the training subset $\mathcal{D}^{\text{train}} = \{\mathcal{S}_1, ..., \mathcal{S}_m, ..., \mathcal{S}_M\}$ and test subset $\mathcal{D}^{\text{test}} = \{\mathcal{S}_1, ..., \mathcal{S}_n, ..., \mathcal{S}_N\}$, where $\mathcal{S}_m \in \mathbb{R}^{T_m \times d_\mathcal{D}}$ and $\mathcal{S}_n \in \mathbb{R}^{T_n \times d_\mathcal{D}}$. Here, T_m and T_n are the number of time steps in the training sequence $\mathcal{S}_m$ and test sequence $\mathcal{S}_n$, respectively, and $d_\mathcal{D}$ the dimensionality of the dataset, i.e. the number of features. For each test sequence $\mathcal{S}_n$ an anomaly score $\mathbf{s}_n$ is assigned which, in an online setting, is a function of time, such that $\mathbf{s}_n \in \mathbb{R}^{T_n}$. Generally, anomalous behaviour is predicted when time steps in $\mathbf{s}_n$ exceed a threshold τ. As explained in Chapter 4.1 the time series data is transformed from its natural variable-length form to fixed length sequences, called *windows*, to facilitate a more efficient and effective training procedure of model-based approaches. This process also needs to be applied to testing, where each sequence is evaluated individually. Hence, an arbitrary sequence $\mathcal{S}_n$ from $\mathcal{D}^{\text{test}}$ is windowed using the `window` function from Algorithm 1, yielding a rank 3 window tensor $\mathbf{X}_n \in \mathbb{R}^{\lambda_n \times w \times d_\mathcal{D}}$, where λ_n represents the number of windows resulting from $\mathcal{S}_n$, w the window size and $d_\mathcal{D}$ the feature count of the dataset $\mathcal{D}$. λ_n is a function of the number of time steps T_n in $\mathcal{S}_n$, the window size w and the window shift k, as shown in Equation 5.1.

$$\lambda_n = (T_n - w)/k + 1 \tag{5.1}$$

Window $\mathbf{W}$ is hence an arbitrary slice of the first axis in $\mathbf{X}_n$, such that $\mathbf{W} \in \mathbb{R}^{w \times d_\mathcal{D}}$.

5.2 Proposed Methodology

5.2.1 Temporal Variational Autoencoder

To detect anomalies in multivariate time series data, TeVAE is proposed, which is illustrated in Figure 5.1. During training, the encoder q_ϕ, consisting of BiLSTM layers and parameterised by set ϕ, maps input window $\mathbf{W}$ to a temporal distribution with

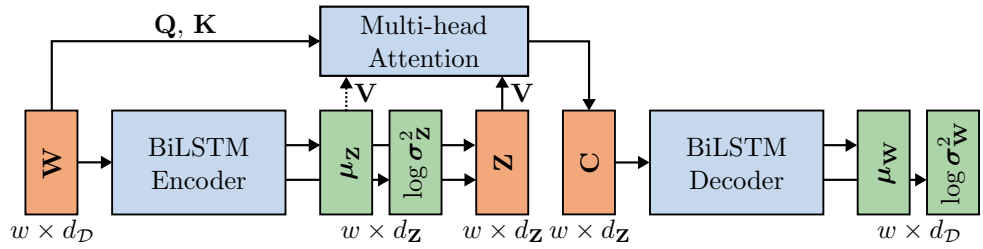

Figure 5.1: An illustration of the proposed TeVAE model. Blue shapes designate trainable models, orange deterministic tensors and green distribution parameters. The shape of each tensor is designated below it. During training $\mathbf{Z}$ is used as the value matrix, denoted by the solid arrow, whereas during inference $\boldsymbol{\mu}_{\mathbf{Z}}$ is used as the value matrix, denoted by the traced arrow.

mean and standard deviation parameters $\boldsymbol{\mu}_{\mathbf{Z}}$ and $\log \boldsymbol{\sigma}_{\mathbf{Z}}^2$, respectively, in the forward pass, Equation 5.2.

$$(\boldsymbol{\mu}_{\mathbf{Z}}, \log \boldsymbol{\sigma}_{\mathbf{Z}}^2) = q_\phi(\mathbf{W}) \tag{5.2}$$

Given the latent distribution with mean and standard deviation parameters $\boldsymbol{\mu}_{\mathbf{Z}}$ and $\log \boldsymbol{\sigma}_{\mathbf{Z}}^2$, respectively, the latent matrix $\mathbf{Z}$ is sampled from the resulting distribution, as shown in Equation 5.3.

$$\mathbf{Z} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{Z}}, \boldsymbol{\sigma}_{\mathbf{Z}}) \tag{5.3}$$

Note that the covariance is not modelled, hence $\boldsymbol{\sigma}_{\mathbf{Z}}$ represents the diagonal of the covariance matrix $\boldsymbol{\Sigma}_{\mathbf{Z}}$, such that $\boldsymbol{\sigma}_{\mathbf{Z}} = \text{diag}(\boldsymbol{\Sigma}_{\mathbf{Z}})$. Then, the input window $\mathbf{W}$ is linearly transformed through weight matrices $\boldsymbol{\Omega}_i^Q$ and $\boldsymbol{\Omega}^{\mathbf{K}}{}_i$ to obtain the query matrices $\mathbf{Q}_i$ and key matrices $\mathbf{K}_i$ for each head i, respectively. Likewise, the sampled latent matrix $\mathbf{Z}$ is also transformed to the value matrix $\mathbf{V}_i$ using weight matrix $\boldsymbol{\Omega}_i^V$, all shown in Equation 5.4.

$$\mathbf{Q}_i = \mathbf{W}\boldsymbol{\Omega}_i^Q \qquad \mathbf{K}_i = \mathbf{W}\boldsymbol{\Omega}_i^K \qquad \mathbf{V}_i = \mathbf{Z}\boldsymbol{\Omega}_i^V \tag{5.4}$$

The weight matrices consist of learnable parameters which are adjusted during the training process. To output the context matrix $\mathbf{C}_i$ for each head i, the softmax of the through $\sqrt{d_{\mathbf{K}}}$ normalised query and key product is multiplied with the value matrix, Equation 5.5.

$$\mathbf{C}_i = \text{Softmax}\left(\frac{\mathbf{Q}_i \mathbf{K}_i^T}{\sqrt{d_{\mathbf{K}}}}\right)\mathbf{V}_i \tag{5.5}$$

The final context matrix $\mathbf{C}$ is the result of the by $\mathbf{\Omega}^O$ linearly transformed concatenation of each head-specific context matrix $\mathbf{C}_i$, as expressed in Equation 5.6.

$$\mathbf{C} = [\mathbf{C}_1, ..., \mathbf{C}_h]\mathbf{\Omega}^O \tag{5.6}$$

The decoder p_θ, also consisting of BiLSTM layers and parameterised by set θ, then maps the context matrix $\mathbf{C}$ to $\boldsymbol{\mu}_\mathbf{W}$ and $\log \boldsymbol{\sigma}^2_\mathbf{W}$, as shown in Equation 5.7. These can then be used to parametrise output distribution $\mathcal{N}(\boldsymbol{\mu}_\mathbf{W}, \boldsymbol{\sigma}_\mathbf{W})$ where, again, covariance is not modelled, i.e. $\boldsymbol{\sigma}_\mathbf{W} = \text{diag}(\mathbf{\Sigma}_\mathbf{W})$.

$$(\boldsymbol{\mu}_\mathbf{W}, \log \boldsymbol{\sigma}^2_\mathbf{W}) = p_\theta(\mathbf{C}) \tag{5.7}$$

Mapping the output to a distribution rather than a deterministic vector allows TeVAE to model uncertainty, which is assumed to be normally distributed. This forward pass is also summarised in Algorithm 2.

Algorithm 2 TeVAE Training Forward Pass

Require: Window $\mathbf{W} \in \mathbb{R}^{w \times d_\mathcal{D}}$
1: $(\boldsymbol{\mu}_\mathbf{Z}, \log \boldsymbol{\sigma}^2_\mathbf{Z}) \leftarrow q_\phi(\mathbf{W})$ ▷ Input window into encoder
2: $\mathbf{Z} \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{Z}, \boldsymbol{\sigma}_\mathbf{Z})$ ▷ Sample from latent distribution
3: **for** $i = 1 \rightarrow h$ **do**
4: $\quad \mathbf{Q}_i = \mathbf{W}\mathbf{\Omega}^Q_i$ ▷ Obtain query matrix for head i
5: $\quad \mathbf{K}_i = \mathbf{W}\mathbf{\Omega}^K_i$ ▷ Obtain key matrix for head i
6: $\quad \mathbf{V}_i = \mathbf{Z}\mathbf{\Omega}^V_i$ ▷ Obtain value matrix for head i
7: $\quad \mathbf{C}_i = \text{Softmax}\left(\frac{\mathbf{Q}_i\mathbf{K}_i^T}{\sqrt{d_\mathbf{K}}}\right)\mathbf{V}_i$ ▷ Obtain context matrix for head i
8: **end for**
9: $\mathbf{C} = [\mathbf{C}_1, ..., \mathbf{C}_h]\mathbf{\Omega}^O$ ▷ Concatenate and transform context matrix
10: $(\boldsymbol{\mu}_\mathbf{W}, \log \boldsymbol{\sigma}^2_\mathbf{W}) \leftarrow p_\theta(\mathbf{C})$ ▷ Input context matrix into decoder
11: **return** Parameters of output distribution $(\boldsymbol{\mu}_\mathbf{W}, \log \boldsymbol{\sigma}^2_\mathbf{W})$

Despite the generative capabilities of variational autoencoders (VAEs), TeVAE does not leverage generation for anomaly detection. Rather than sampling a latent matrix $\mathbf{Z}$ as shown in Equation 5.3 during inference, sampling is disabled and only $\boldsymbol{\mu}_\mathbf{Z}$ is taken as the input for the multi-head attention mechanism, as done by von Schleinitz et al. [80]. Equation 5.3 (line 2 in Algorithm 2), therefore, is replaced by Equation 5.8 for the forward pass, as shown in line 2 in Algorithm 3.

$$\mathbf{Z} = \boldsymbol{\mu}_\mathbf{Z} \tag{5.8}$$

This not only accelerates inference by eliminating the sampling process but is also

Algorithm 3 TeVAE Inference Forward Pass (`tevae`)

Require: Window $\mathbf{W} \in \mathbb{R}^{w \times d_{\mathcal{D}}}$
$(\boldsymbol{\mu}_{\mathbf{Z}}, \log \boldsymbol{\sigma}^2_{\mathbf{Z}}) \leftarrow q_\phi(\mathbf{W})$ ▷ Input window into encoder
$\mathbf{Z} = \boldsymbol{\mu}_{\mathbf{Z}}$ ▷ Instead of sampling from $\mathcal{N}(\boldsymbol{\mu}_{\mathbf{Z}}, \boldsymbol{\sigma}_{\mathbf{Z}})$
for $i = 1 \rightarrow h$ **do**
  $\mathbf{Q}_i = \mathbf{W}\boldsymbol{\Omega}_i^Q$ ▷ Obtain query matrix for head i
  $\mathbf{K}_i = \mathbf{W}\boldsymbol{\Omega}_i^K$ ▷ Obtain key matrix for head i
  $\mathbf{V}_i = \mathbf{Z}\boldsymbol{\Omega}_i^V$ ▷ Obtain value matrix for head i
  $\mathbf{C}_i = \text{Softmax}\left(\frac{\mathbf{Q}_i \mathbf{K}_i^T}{\sqrt{d_{\mathbf{K}}}}\right)\mathbf{V}_i$ ▷ Obtain context matrix for head i
end for
$\mathbf{C} = [\mathbf{C}_1, ..., \mathbf{C}_h]\boldsymbol{\Omega}^O$ ▷ Concatenate and transform context matrix
$(\boldsymbol{\mu}_{\mathbf{W}}, \log \boldsymbol{\sigma}^2_{\mathbf{W}}) \leftarrow p_\theta(\mathbf{C})$ ▷ Input context matrix into decoder
return Parameters of output distribution $(\boldsymbol{\mu}_{\mathbf{W}}, \log \boldsymbol{\sigma}^2_{\mathbf{W}})$

empirically found to be a good approximation of an averaged latent matrix if it were sampled several times as done by Pereira et al. [100]. The TeVAE layout during inference is also shown in Figure 5.1, where the traced arrow designates the information flow from the encoder to the multi-head attention (MA) mechanism during inference.

5.2.2 Bypass Phenomenon

VAEs, when combined with an attention mechanism, can exhibit a behaviour called the bypass phenomenon [99]. When the bypass phenomenon occurs, the latent path between encoder and decoder is bypassed, and information flow occurs mostly or exclusively through the attention mechanism, as it has deterministic access to the encoder hidden states and therefore avoids regularisation through the D_{KL} term. In an attempt to counteract this, Bahuleyan et al. [99] propose variational attention, which, like the VAE, maps the input to a distribution rather than a deterministic vector. Applied to natural language processing, Bahuleyan et al. [99] demonstrate that this leads to a diversified generated portfolio of sentences, indicating alleviation of the bypassing phenomenon. Only Pereira et al. [100] apply this insight in the anomaly detection domain; however, they do not present any evidence of alleviation of the bypass phenomenon in their work. TeVAE on the other hand, cannot suffer from the bypass phenomenon in the sense that information flow ignores the latent variational path between encoder and decoder, since the MA mechanism requires the value matrix $\mathbf{V}$ from the encoder to output the context matrix. Assuming the bypass phenomenon also applies to a case where information flow ignores the attention mechanism, one

could claim that TeVAE is not immune. To investigate this further, the attention mechanism is removed from the model in an ablation study to see if anomaly detection performance is affected. In this case, $\mathbf{Z}$ is instead directly input into the decoder during training, whereas $\mu_{\mathbf{Z}}$ is used during inference. If detection performance drops, it is considered evidence of the contribution of the attention mechanism to the model performance and hence is not bypassed. The results of this ablation study are shown and discussed in Chapter 5.3.

5.2.3 Mapping Fixed-length Windows to Variable-length Sequences

Since the TeVAE is trained to reconstruct fixed-length windows, the same applies during inference. However, to decide whether a given measurement sequence $\mathcal{S}_n \in \mathbb{R}^{T_n \times d_{\mathcal{D}}}$ is anomalous, a continuous reconstruction of the measurement is required. A trivial way to do so would be to window the input measurement $\mathcal{S}_n$ using a shift $k = 1$, input the windows into the model and append the last time step from each output window to obtain a continuous sequence [101], which is referred to as last-type RWM from now on. Alternatively, it is also possible to invert the same procedure by taking the first time step rather than the last one, which from now on is referred to as first-type RWM. Considering the BiLSTM nature of the encoder and decoder, the first and last time steps of a window can only be computed given the states from one direction, making these values, in theory, less accurate. To overcome this, averaging matching time steps in overlapping windows is proposed, which is called *mean-type* RWM. In practice, overlapping time steps of different windows are added together, while a counter array keeps track of how many windows contribute to each time step. Finally, each time step is divided by the corresponding value inside the counter array, yielding the average, as detailed in Algorithm 4.

With the foundations laid out, the process of inference and detecting anomalies can be introduced. The input for this process is the output distribution parameters $(\boldsymbol{\mu}_{\mathbf{W}}, \log \boldsymbol{\sigma}^2_{\mathbf{W}})$, so it essentially averages the distributions and hence can only be applied to the mean and *variance* parameters. Consider the distributions $\mathcal{N}(\mu_x, \sigma_x^2)$ and $\mathcal{N}(\mu_y, \sigma_y^2)$, both assumed to be independent and normally distributed. The sum of both distributions results in normal distribution $\mathcal{N}(\mu_{x+y}, \sigma^2_{x+y})$, obtained as shown in Equation 5.9 [102].

$$\mu_{x+y} = \mu_x + \mu_y \qquad \sigma^2_{x+y} = \sigma^2_x + \sigma^2_y \tag{5.9}$$

Therefore, the standard deviation of the resulting distribution σ_{x+y} is characterised

Algorithm 4 Mean-type RWM (`reverse_window`)

Require: Window Tensor $\mathbf{X} \in \mathbb{R}^{\lambda \times w \times d_{\mathcal{D}}}$, Window Shift $k \in \mathbb{N}$

$\hat{\mathcal{S}}_n \leftarrow \text{zeros}(T_n, d_{\mathcal{D}})$ ▷ Pre-allocate sequence
$\mathbf{c} \leftarrow \text{zeros}(T_n, d_{\mathcal{D}})$ ▷ Pre-allocate counter array
for $i = 1 \rightarrow \lambda$ **do**
  $\mathbf{W} \leftarrow \mathbf{X}[i]$ ▷ Get one window from the window tensor
  $\hat{\mathcal{S}}_n[i * k : i * k + w] \leftarrow \hat{\mathcal{S}}_n[i * k : i * k + w] + \mathbf{W}$ ▷ Add window
  $\mathbf{c}[i * k : i * k + w] \leftarrow \mathbf{c}[i * k : i * k + w] + 1$ ▷ Update counter
end for
$\hat{\mathcal{S}}_n \leftarrow \hat{\mathcal{S}}_n / \mathbf{c}$ ▷ Obtain average by dividing by counter
return Sequence $\hat{\mathcal{S}}_n \in \mathbb{R}^{T_n \times d_{\mathcal{D}}}$

by Equation 5.10.

$$\sigma_{x+y} = \sqrt{\sigma_x^2 + \sigma_y^2} \tag{5.10}$$

Hence, to take the mean of the distributions, the variance $\boldsymbol{\sigma}_{\mathbf{W}}^2$ is averaged, and then converted to the *standard deviation* $\boldsymbol{\sigma}_{\mathbf{W}}$.

With a continuous mean $\boldsymbol{\mu}_{\mathcal{S}}$ and standard deviation $\boldsymbol{\sigma}_{\mathcal{S}}$, the continuous negative log-likelihood, i.e. the anomaly score $\mathbf{s}$, is computed for the respective sequence. The anomaly detection process is also outlined in Algorithm 5. Note that, this algorithm makes use of the `window` function from Algorithm 1, the TeVAE inference forward pass function `tevae` function from Algorithm 3 and the `reverse_window` function from Algorithm 4.

Algorithm 5 Anomaly Detection Process

Require: Sequence $\mathcal{S}_n \in \mathbb{R}^{T_n \times d_{\mathcal{D}}}$, Threshold $\tau \in \mathbb{R}$

$\mathbf{X} \leftarrow \texttt{window}(\mathcal{S}_n)$ ▷ Obtain window tensor from sequence
for $i = 1 \rightarrow \lambda$ **do**
  $\mathbf{W} \leftarrow \mathbf{X}[i]$ ▷ Get one window from the window tensor
  $(\boldsymbol{\mu}_{\mathbf{W}}, \log \boldsymbol{\sigma}_{\mathbf{W}}^2) \leftarrow \texttt{tevae}(\mathbf{W})$ ▷ Inference forward pass by TeVAE
  $\boldsymbol{\mu}_{\mathbf{X}}[i] \leftarrow \boldsymbol{\mu}_{\mathbf{W}}$
  $\boldsymbol{\sigma}_{\mathbf{X}}^2[i] \leftarrow \boldsymbol{\sigma}_{\mathbf{W}}^2$
end for
$\boldsymbol{\mu}_{\mathcal{S}} \leftarrow \texttt{reverse\_window}(\boldsymbol{\mu}_{\mathbf{X}})$ ▷ Reverse-window output mean parameter
$\boldsymbol{\sigma}_{\mathcal{S}}^2 \leftarrow \texttt{reverse\_window}(\boldsymbol{\sigma}_{\mathbf{X}}^2)$ ▷ Reverse-window output variance parameter
$\mathbf{s}_n \leftarrow -\log p(\boldsymbol{\mu}_{\mathcal{S}}, \boldsymbol{\sigma}_{\mathcal{S}} | \mathcal{S}_n)$ ▷ Obtain continuous anomaly score
$l_n \leftarrow \max(\mathbf{s}_n) > \tau$ ▷ Compare maximum value in anomaly score with threshold
return Label l_n for sequence $\mathcal{S}_n$

A comparison between the detection performance obtained using the mean-type,

last-type, and first-type RWMs is provided in Chapter 5.3.

The intrinsic delay δ_{intr} associated with each of the RWMs can be discussed ahead of the results in Chapter 5.3, however. δ_{intr} is defined as the delay naturally introduced by each RWM. To aid analysis, the intrinsic delay δ_{intr} is plotted against time t to demonstrate the delay for each of the RWMs, shown in Figure 5.2.

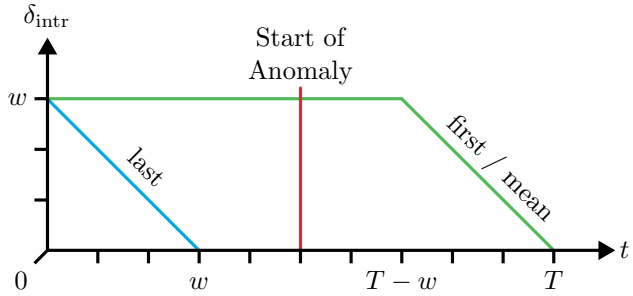

Figure 5.2: Intrinsic delay δ_{intr} plotted against time t. The last-type RWM is represented by a solid blue line, and the first-type and mean-type by a solid green line. The red line represents the start of a hypothetical sub-sequence anomaly.

Consider the last-type RWM during $0 < t < w$. Since the models considered in this thesis are trained on fixed-length windows, evaluation can only start once w time steps are recorded to constitute a window, meaning that an anomalous subsequence starting at $t = 0$ can only be detected $\delta_{\text{intr}} = w$ time steps later. Likewise, an anomaly starting at $t = 1$ can only be detected $\delta_{\text{intr}} = w - 1$ time steps later and another starting at $t = w$ can be detected immediately, which holds true for the rest of the sequence, as the last time step of each window corresponds to the current real-world time step, i.e. $\delta_{\text{intr}} = 0$ for $w < t < T$. A non-zero intrinsic delay until $t = w$ is natural to any window-based approach, and hence the first-type and mean-type RWMs show the same behaviour. In contrast to last-type RWM, however, these methods only output the first value of each window, i.e. evaluation is $\delta_{\text{intr}} = w$ time steps behind for $0 < t < T - w$. At $t = T$, though, the time steps $T - w < t < T$ are all output at the same time, meaning that $\delta_{\text{intr}} = 0$ and no extra time is needed for evaluation after streaming ends.

It becomes clear that, in *online* time series anomaly detection, the last-type RWM is in theory faster than the other two types. For instance, consider a sub-sequence anomaly starting at time step $w + 2$, designated by the red line in Figure 5.2. Apart from the time required for inference, the last-type RWM can detect the anomaly with a intrinsic delay $\delta_{\text{intr}} = 0$, whereas the other two methods can only detect the anomaly

$\delta_{\text{intr}} = w$ time steps later. For $0 < t < T - w$, a intrinsic delay of $\delta_{\text{intr}} = w$ time steps is, therefore, the best that first-type and mean-type RWMs can achieve, however, while the best delay the last-type can achieve is $\delta_{\text{intr}} = 0$, it will be higher in reality, perhaps higher than w time steps.

5.2.4 Generalisation Capabilities

In some real-world cases, the available training data may not cover all real-world driving scenarios, for example at the initial stages of a project with limited data availability, or outside a test bench scenario on a real road, where driving does not follow a provided pattern, but is much more random due to external influences, like traffic or weather. Translating this to an experimental setting, some driving cycles within the test subset are not present in the training subset, unlike the previous experiments. Modelling nominal behaviour is therefore especially challenging, as the main problem involves distinguishing between nominal but unseen driving patterns and anomalous driving patterns. To investigate how well TeVAE performs in such a setting, the powertrain anomaly time series benchmark (PATH) training subset is split into two sets of three folds. Driving cycles are characterised by the vehicle speed, but it is not available as a channel, however vehicle speed is directly proportional to axle angular speed, and hence it is used as a proxy instead. The first set of folds consists of sequences with a low, medium, and high average axle angular speed, essentially splitting by operational range. Splitting the dataset this way will lead to the folds representing mostly urban, country roads and motorway rides, respectively. The second set of folds, on the other hand, consists of sequences with a low, medium, and high axle angular speed standard deviation, therefore splitting by levels of dynamics. Splitting the dataset this way will lead to the folds representing mostly constant, mixed and highly dynamic rides, respectively. For each fold, TeVAE is trained using seeds 1, 2 and 3, and is evaluated using the same test subset each time.

5.2.5 Experimental Setup

Threshold Estimation Method

As mentioned in Chapter 3.2, the vast majority of literature chooses a detection threshold τ by using labelled data one way or another, despite claiming to propose an unsupervised approach to anomaly detection. Therefore, to provide a truly unsupervised baseline, the so-called *unsupervised threshold* τ_{us} is set as the maximum anomaly score

observed in the validation subset $\mathcal{D}^{\text{val}}$. In cases with exclusively nominal sequences in $\mathcal{D}^{\text{val}}$, this will naturally lead to a high precision, as the number of false positives $N_{\mathbf{fp}}$ will be low. In cases where $\mathcal{D}^{\text{val}}$ may be contaminated by anomalous data, however, this threshold choice can lead to poor detection performance. For instance, if, due to a high anomaly score from an anomaly in $\mathcal{D}^{\text{val}}$, threshold τ is set too large, all sequences within $\mathcal{D}^{\text{test}}$ will be labelled as nominal, leading to a perfect precision figure of $P = 1$ but a recall of $R = 0$ and therefore an $F_1 = 0$.

Clearly, the threshold choice has a large impact on the anomaly detection performance, though it is not the only factor, which can hold back the detection performance of an otherwise well-performing approach. To ablate the threshold choice from other factors that influence the detection performance, the so-called *hypothetical best threshold* τ_{best} is employed. τ_{best} is the threshold that leads to the best F_1 score on the respective test subset possible with the respective approach, and is obtained by grid-searching through the percentiles of all anomaly scores stemming from the test subset $\mathcal{D}^{\text{test}}$. Clearly, this would not be observable in the real world given the absence of a labelled test subset, hence why it is referred to as *hypothetical*.

Evaluation Metrics

As discussed in Chapter 3.1, no set of evaluation metrics proposed in the literature takes timeliness into account, is parameter-free and compatible with discrete-sequence problems. To address this, this section introduces a novel set of metrics. These metrics mostly rely on the conventional definition of true positive, false positive, true negative and false negative labels applied to each individual discrete sequence, except in the case of a sub-sequence anomaly.

For partially anomalous time series, it can be labelled as a true positive or a false negative, but also as a false positive, which occurs when an anomaly detector flags a time step prematurely. Formally, this is the case when the first flagged time step is far enough ahead of the first ground-truth time step that the model cannot have had access to it. As is evident, the proposed metrics can only be applied to discrete time series data sets where anomalous time series have at most one contiguous ground-truth anomalous sub-sequence of any length.

In addition to that, the time between detection and ground-truth anomaly start is also quantified for each anomalous $\mathcal{S}_n$, with false positives being assigned the absolute value of the "negative" delay and false negatives being assigned the length of the anomalous sub-sequence within $\mathcal{S}_n$. The detection delays are finally aggregated into the average detection delay $\bar{\delta}$, given in seconds. The issue with these metrics is that

they are not *recall consistent* as defined by Wagner et al. [33], meaning that the recall monotonically decreases with an increasing threshold. Consider a time series with a sub-sequence anomaly starting off as nominal and eventually becoming anomalous, as shown in Figure 5.3. For a very low threshold τ_1, the anomaly is considered a false

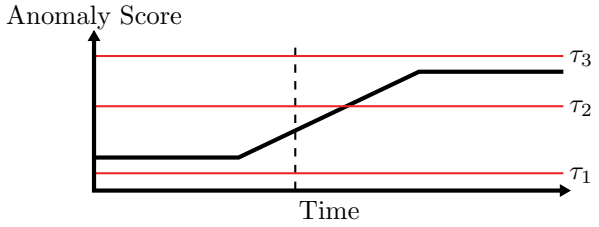

Figure 5.3: Arbitrary anomaly score as a function of time. The dotted vertical line denotes the start of the anomalous behaviour, and the red lines represent three different thresholds.

positive, since it is an early detection. As the threshold increases to τ_2, it leads to a true positive, but when the threshold reaches τ_3, it becomes a false negative. This leads to an increase and then decrease in recall, and hence the metrics do not qualify as recall consistent, which also prevents them from being used to calculate the area under the precision-recall curve to quantify the uncalibrated detection performance. Additionally, unlike the improved range-based recall R_{rb} and precision P_{rb} by Wagner et al. [33], this set of metrics does not account for predictive patterns. Despite its short-comings, it is the only set of metrics that are apt for online discrete-sequence problems. Nonetheless, there is still a clear need for a novel metric which combines the compatibility with discrete-sequence problems with recall consistency and consideration of predictive patterns.

Competing Approaches

In this work, several approaches from the literature are considered, which either published the source code provided enough information for re-implementation. The selection includes various autoencoder-based models, as well as a state-of-the-art nonparametric approach.

In the time series anomaly detection literature, the only other model that uses the combination of a VAE and an attention mechanism is by Pereira et al. [100]. For the purpose of this work, it is referred to as variational self-attention VAE (VS-VAE). Their approach consists of a BiLSTM encoder and decoder, where, for an input window of length w, the $t = w$ encoder hidden states of each direction are passed on to the variational self-attention (VS) mechanism [99]. The resulting context vector is

then concatenated with the sampled latent vector and then passed on to the decoder. Pereira et al. [100] claims that applying VS to the VAE solves the bypass phenomenon, however, no evidence for this claim is provided.

OmniAnomaly (OA) [2] attempts to create a temporal connection between latent distributions by applying a linear Gaussian state space model to them. Furthermore, it concatenates the last gated recurrent unit (GRU) hidden state with the latent vector sampled in the previous time step. In addition to that, it uses planar normalising flow [103] by applying K transformations to the latent vector to approximate a non-Gaussian posterior.

As part of the VAE-based selective prediction (VASP) framework [80], a variational autoencoder architecture is proposed to increase the robustness of time series prediction when faced with anomalies. While the main contribution is attributed to the framework itself, not the VAE, it should be noted that during inference only the mean parameter of the latent distribution is passed to the decoder.

Thill et al. [71] suggest a temporal convolutional autoencoder (TCN-AE) with skip connections to prevent an overreliance of the model on the high dilation rate representation. Furthermore, Thill et al. propose using the reduced representations between hidden layers in addition to map reduction layers after each hidden layer to reduce the dimensionality of the convolution and therefore the number of tuneable parameters.

Smoothness-inducing sequential VAE (SISVAE) [81] tries to improve the modelling robustness by the addition of a smoothing term in the loss function which contributes to the reduction of sudden changes in the reconstructed signal, making it less sensitive to noisy time steps.

The lightweight LSTM-VAE (LW-VAE) [94] is another variational autoencoder model which offers no significant contributions besides being relatively parameter-light.

Time series anomaly detection through incremental search (TSADIS) [104] is a non-parametric approach based on the matrix profile (MP) computation. As far as the author is aware, it is the only MP-based approach that can be applied to multivariate time series problems. One major benefit of TSADIS, is that, given its non-parametric nature, it does not rely on a computationally expensive training procedure, lowering the hurdle of implementation in the real world. It is simply called on the entire variable-length sequence, which while simple, also means that it is technically an *offline* approach, as all time steps need to be recorded before it can be evaluated.

The hyperparameters for each approach are set as specified in the respective publi-

cation, though early stopping is applied to all that require a training procedure. Early stopping is parameterised such that the respective reconstruction error is monitored and training is stopped once validation loss has stopped decreasing for 250 epochs. To this end, 20 % of the unlabelled training subset $\mathcal{D}^{\text{train}}$ is separated to form the validation subset $\mathcal{D}^{\text{val}}$.

Environment

The framework used for model training is TensorFlow 2.15.1 and TensorFlow Probability 0.23 on Python 3.10 on a workstation running Ubuntu 22.04.5 LTS, equipped with two Nvidia RTX A6000 GPUs. All work involving TSADIS is done in a separate environment with the latest compatible Python version of 3.9. Further information on library versions used can be found in the *requirements.txt* file in the GitHub repository under `github.com/lcs-crr/Thesis`.

5.3 Results

5.3.1 Online Anomaly Detection

PATH Dataset

First, the selection of approaches is tested on the unsupervised version of the PATH dataset. The PATH dataset is a synthetic but real-world-inspired dataset, representing the behaviour of an automotive powertrain. For more details, please refer to Chapter 4.1. This version of the dataset has a contaminated training subset, i.e. it contains unlabelled anomalies. The results obtained using the unsupervised threshold τ_{us} and the hypothetical best threshold τ_{best} are shown in tabular form in Table 5.1.

All approaches perform poorly in terms of F_1 score when τ_{us} is employed, with OA performing slightly better. Over all the folds and seeds, the average precision P varies by a large margin. On average, VS-VAE manages to label 78 % of the ground-truth nominal sequences as nominal, with OA 60 % and TCN-AE 48 %. When it comes to recall R, however, all approaches achieve low figures, which explains the high average detection delay $\bar{\delta}$ and low F_1 scores. Additionally, TSADIS finds no anomalies, which is why F_1 score, precision P and recall R are 0 throughout. These observations can be explained by the threshold choice, which sets the boundary between nominal and anomalous samples too large, sometimes even so high, that all sequences in $\mathcal{D}^{\text{test}}$ lie below it, as is the case with TSADIS. After finding τ_{best}, however, a poor threshold

Table 5.1: Mean ± standard deviation of F_1 score, precision P, recall R, and average detection delay $\bar{\delta}$ using the unsupervised threshold τ_{us} (top half) and hypothetical best threshold τ_{best} (bottom half) for a range of approaches applied to the *unsupervised* anomaly detection version of the PATH dataset, i.e. the version with anomalous data in training subset $\mathcal{D}^{\text{train}}$. The best F_1 scores are provided in **bold**.

Approach	F_1	P	R	$\bar{\delta}$ $[s]$	
VS-VAE	0.03 ± 0.02	0.78 ± 0.42	0.01 ± 0.01	991.4 ± 71.1	
OA	$\mathbf{0.06 \pm 0.06}$	0.60 ± 0.44	0.03 ± 0.03	994.9 ± 69.7	
VASP	0.02 ± 0.02	0.21 ± 0.34	0.01 ± 0.01	991.9 ± 70.8	
TCN-AE	0.02 ± 0.02	0.45 ± 0.41	0.01 ± 0.01	989.8 ± 66.9	τ_{us}
SISVAE	0.03 ± 0.03	0.13 ± 0.10	0.02 ± 0.02	993.1 ± 69.6	
LW-VAE	0.01 ± 0.02	0.26 ± 0.41	0.01 ± 0.01	991.9 ± 71.0	
TSADIS	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	–	
TeVAE	0.02 ± 0.03	0.27 ± 0.36	0.01 ± 0.02	992.7 ± 70.7	
VS-VAE	0.30 ± 0.07	0.35 ± 0.26	0.34 ± 0.05	792.0 ± 75.4	
OA	0.50 ± 0.09	0.60 ± 0.11	0.44 ± 0.11	767.0 ± 114.7	
VASP	0.11 ± 0.01	0.07 ± 0.01	0.47 ± 0.24	673.5 ± 181.2	
TCN-AE	0.14 ± 0.01	0.09 ± 0.01	0.41 ± 0.16	750.2 ± 121.1	τ_{best}
SISVAE	0.22 ± 0.03	0.19 ± 0.05	0.28 ± 0.04	826.2 ± 80.0	
LW-VAE	0.13 ± 0.01	0.07 ± 0.01	0.51 ± 0.22	634.8 ± 172.6	
TSADIS	0.10 ± 0.01	0.05 ± 0.00	1.00 ± 0.00	–	
TeVAE	$\mathbf{0.58 \pm 0.08}$	0.69 ± 0.12	0.50 ± 0.09	676.4 ± 103.2	

choice can be isolated from the detection performance. Now, a larger gap between F_1 scores is observable, with TeVAE performing the best, with a score of $F_1 = 0.58$ and a much reduced average detection delay of $\bar{\delta} = 676.4\,\text{s}$. OA and VS-VAE perform second and third best, scoring $F_1 = 0.50$ and $F_1 = 0.30$, respectively.

Next, the selection of approaches is tested on the semi-supervised version of the PATH dataset. This version of the dataset consists of a nominal-only training subset, i.e. it contains no anomalies. The results obtained using the unsupervised threshold τ_{us} and the hypothetical best threshold τ_{best} are shown in tabular form in Table 5.2.

For the semi-supervised results using τ_{us}, the detection performance of some approaches improve significantly, namely for VS-VAE, OA, SISVAE and TeVAE. All above-mentioned approaches now label over 90 % of the nominal sequences as such. At least for TeVAE, the average detection delay has now also decreased by 41 %. These gains can be attributed to a much better placement of the threshold, as well as more successful modelling of nominal behaviour without contamination within the training subset. Despite the relaxed conditions in this experiment compared to the conditions

Table 5.2: Mean ± standard deviation of F_1 score, precision P, recall R, and average detection delay $\bar{\delta}$ using the unsupervised threshold τ_{us} (top half) and hypothetical best threshold τ_{best} (bottom half) for a range of approaches applied to the *semi-supervised* anomaly detection version of the PATH dataset, i.e. the version without anomalous data in training subset $\mathcal{D}^{\text{train}}$. The best F_1 scores are provided in **bold**.

	Approach	F_1	P	R	$\bar{\delta}$ $[s]$
τ_{us}	VS-VAE	0.43 ± 0.15	0.90 ± 0.09	0.30 ± 0.11	825.9 ± 130.8
	OA	$\mathbf{0.67 \pm 0.15}$	0.96 ± 0.04	0.54 ± 0.19	713.7 ± 203.8
	VASP	0.04 ± 0.03	0.34 ± 0.28	0.02 ± 0.01	986.7 ± 64.7
	TCN-AE	0.02 ± 0.02	0.37 ± 0.39	0.01 ± 0.01	988.6 ± 66.6
	SISVAE	0.61 ± 0.09	0.96 ± 0.05	0.46 ± 0.11	807.8 ± 151.8
	LW-VAE	0.03 ± 0.02	0.40 ± 0.35	0.02 ± 0.01	987.5 ± 66.0
	TSADIS	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	–
	TeVAE	0.63 ± 0.22	0.95 ± 0.06	0.52 ± 0.26	589.1 ± 212.0
τ_{best}	VS-VAE	0.53 ± 0.05	0.84 ± 0.14	0.40 ± 0.07	780.1 ± 100.0
	OA	$\mathbf{0.86 \pm 0.05}$	0.88 ± 0.05	0.85 ± 0.06	467.7 ± 91.6
	VASP	0.12 ± 0.01	0.07 ± 0.01	0.42 ± 0.19	712.6 ± 148.7
	TCN-AE	0.15 ± 0.01	0.09 ± 0.01	0.48 ± 0.12	703.0 ± 96.1
	SISVAE	0.71 ± 0.05	0.84 ± 0.10	0.62 ± 0.04	732.6 ± 124.2
	LW-VAE	0.13 ± 0.01	0.08 ± 0.01	0.48 ± 0.23	655.8 ± 162.6
	TSADIS	0.10 ± 0.01	0.05 ± 0.00	1.00 ± 0.00	–
	TeVAE	0.80 ± 0.15	0.88 ± 0.11	0.76 ± 0.18	412.6 ± 143.8

using the contaminated training subset, VASP, TCN-AE and LW-VAE still fail to yield satisfactory results, all scoring less than $F_1 = 0.05$. Looking at the semi-supervised results using τ_{best} reveals these approaches do not only struggle due to the threshold choice, but also with the challenge of detecting anomalies in this dataset in general. While their results have slightly improved, none score more than $F_1 = 0.15$. In contrast, VS-VAE, OA, SISVAE and TeVAE make slight gains, with OA again scoring the highest with $F_1 = 0.86$. Because TSADIS does not rely on a training procedure, its results naturally remain the same, regardless of the presence of contamination in $\mathcal{D}^{\text{train}}$.

Proprietary Dataset

Now, the selection of approaches is tested on a real-world dataset. This dataset also represents the behaviour of an automotive powertrain on a test bench in a semi-supervised setting, meaning that the training subset $\mathcal{D}^{\text{train}}$ consists of nominal sequences only. For more details, please refer to Chapter 4.2. The results obtained

using the unsupervised threshold τ_{us} and the hypothetical best threshold τ_{best} are shown in tabular form in Table 5.3.

Table 5.3: Mean ± standard deviation of F_1 score, precision P, recall R, and average detection delay $\bar{\delta}$ using the unsupervised threshold τ_{us} (top half) and hypothetical best threshold τ_{best} (bottom half) for a range of approaches applied to the real-world dataset. The best F_1 scores are provided in **bold**.

Approach	F_1	P	R	$\bar{\delta}$ $[s]$	
VS-VAE	0.41 ± 0.19	0.86 ± 0.07	0.30 ± 0.19	697.9 ± 120.4	
OA	0.26 ± 0.07	0.67 ± 0.16	0.17 ± 0.05	782.9 ± 19.2	
VASP	0.14 ± 0.04	0.66 ± 0.11	0.08 ± 0.02	792.1 ± 6.6	
TCN-AE	0.24 ± 0.03	0.79 ± 0.11	0.14 ± 0.02	773.6 ± 5.9	τ_{us}
SISVAE	0.35 ± 0.12	0.73 ± 0.16	0.25 ± 0.14	742.7 ± 78.9	
LW-VAE	0.16 ± 0.03	0.75 ± 0.07	0.09 ± 0.02	791.1 ± 6.2	
TSADIS	0.11 ± 0.00	0.37 ± 0.04	0.06 ± 0.00	–	
TeVAE	$\mathbf{0.42 \pm 0.16}$	0.83 ± 0.12	0.31 ± 0.14	720.5 ± 59.3	
VS-VAE	0.53 ± 0.15	0.65 ± 0.24	0.55 ± 0.21	569.1 ± 147.8	
OA	0.38 ± 0.03	0.42 ± 0.24	0.59 ± 0.29	650.1 ± 121.2	
VASP	0.43 ± 0.03	0.43 ± 0.04	0.44 ± 0.05	546.1 ± 46.6	
TCN-AE	0.52 ± 0.03	0.57 ± 0.10	0.49 ± 0.05	622.7 ± 46.0	τ_{best}
SISVAE	0.42 ± 0.09	0.61 ± 0.30	0.54 ± 0.30	609.0 ± 169.4	
LW-VAE	0.37 ± 0.05	0.39 ± 0.08	0.38 ± 0.07	636.4 ± 55.3	
TSADIS	0.12 ± 0.00	0.07 ± 0.00	1.00 ± 0.00	–	
TeVAE	$\mathbf{0.64 \pm 0.10}$	0.82 ± 0.15	0.55 ± 0.12	642.5 ± 67.9	

The results indicate that this dataset is more challenging compared to the semi-supervised version of the PATH dataset. When τ_{us} is used, TeVAE and VS-VAE perform best and identically relative to the other approaches. While OA, SISVAE and TeVAE perform considerably worse than with the semi-supervised PATH, VS-VAE remains remarkably consistent. VASP, TCN-AE, LW-VAE and TSADIS, on the other hand, actually improve their results on the real-world dataset. Using τ_{best} closes the field significantly. If TSADIS is ignored, the results for the semi-supervised PATH dataset using τ_{best} lie anywhere between $F_1 = 0.12$ and $F_1 = 0.86$, whereas in the case of the real-world dataset between $F_1 = 0.37$ and $F_1 = 0.64$. Furthermore, using τ_{best} leads to modest detection performance improvements across the board, except for TSADIS. TSADIS performs similarly in both the PATH and the real-world dataset with τ_{best}, achieving perfect recall R but poor precision P both times, suggesting that τ_{best} is set very low.

Like with the PATH dataset, these results also indicate that a more refined thresh-

old choice can improve the detection performance.

5.3.2 Ablation Study for Bypass Phenomenon

To find the impact of multi-head attention on TeVAE, its absence is tested on the unsupervised and semi-supervised versions of the PATH dataset. The results obtained are shown in tabular form in Tables 5.4 and 5.5.

Table 5.4: Mean ± standard deviation of F_1 score, precision P, recall R, and average detection delay $\bar{\delta}$ using the unsupervised threshold τ_{us} (top half) and hypothetical best threshold τ_{best} (bottom half) for TeVAE with and without the multi-head attention mechanism (NoMA) applied to the *unsupervised* anomaly detection version of the PATH dataset, i.e. the version with anomalous data in training subset $\mathcal{D}^{\text{train}}$. The best F_1 scores are provided in **bold**.

Approach	F_1	P	R	$\bar{\delta}$ $[s]$	
NoMA	$\mathbf{0.03 \pm 0.04}$	0.33 ± 0.38	0.02 ± 0.02	992.7 ± 68.4	τ_{us}
TeVAE	0.02 ± 0.03	0.27 ± 0.36	0.01 ± 0.02	992.7 ± 70.7	
NoMA	0.36 ± 0.09	0.57 ± 0.23	0.28 ± 0.06	817.1 ± 66.1	τ_{best}
TeVAE	$\mathbf{0.58 \pm 0.08}$	0.69 ± 0.12	0.50 ± 0.09	676.4 ± 103.2	

Considering the results previously shown in Table 5.1, it is not surprising that both are essentially useless, with no major discernable difference between TeVAE and NoMA when τ_{us} is used on the unsupervised PATH dataset. Using τ_{best}, however, shows that the presence of the multi-head attention mechanism in TeVAE makes a significant difference. Its absence in NoMA reduces the F_1 score by 38 % and increases the average detection delay by 47 %.

Table 5.5: Mean ± standard deviation of F_1 score, precision P, recall R, and average detection delay $\bar{\delta}$ using the unsupervised threshold τ_{us} (top half) and hypothetical best threshold τ_{best} (bottom half) for TeVAE with and without the multi-head attention mechanism (NoMA) to the *semi-supervised* anomaly detection version of the PATH dataset, i.e. the version with anomalous data in training subset $\mathcal{D}^{\text{train}}$. The best F_1 scores are provided in **bold**.

Approach	F_1	P	R	$\bar{\delta}$ $[s]$	
NoMA	$\mathbf{0.66 \pm 0.20}$	0.96 ± 0.02	0.54 ± 0.24	572.9 ± 193.0	τ_{us}
TeVAE	0.63 ± 0.22	0.95 ± 0.06	0.52 ± 0.26	589.1 ± 212.0	
NoMA	0.77 ± 0.15	0.91 ± 0.07	0.69 ± 0.19	466.9 ± 175.6	τ_{best}
TeVAE	$\mathbf{0.80 \pm 0.15}$	0.88 ± 0.11	0.76 ± 0.18	412.6 ± 143.8	

Applying both variants on the semi-supervised version of the PATH dataset, once again shows that, in the absence of contaminated data, τ_{us} leads to much higher

F_1 scores. Furthermore, it becomes clear that NoMA performs much closer to TeVAE regardless of threshold used, indicating that here, the multi-head attention mechanism plays less of a role. In light of the results using τ_{best} in Table 5.4, however, the multi-head attention mechanism seems to contribute to more robust modelling of nominal behaviour when training subset $\mathcal{D}^{\text{train}}$ is contaminated.

5.3.3 Impact of Reverse-window Method

To measure the potential benefit of the mean-type RWM, it is compared to the first-type and last-type RWMs using both the unsupervised and semi-supervised versions of the PATH dataset. The respective results are shown in Tables 5.6 and 5.7.

Table 5.6: Mean ± standard deviation of F_1 score, precision P, recall R, and average detection delay $\bar{\delta}$ using the unsupervised threshold τ_{us} (top half) and hypothetical best threshold τ_{best} (bottom half) for different types of RWM applied to the *unsupervised* anomaly detection version of the PATH dataset, i.e. the version with anomalous data in training subset $\mathcal{D}^{\text{train}}$. The best F_1 scores are provided in **bold**.

RWM	F_1	P	R	$\bar{\delta}$ $[s]$	
first	0.00 ± 0.01	0.11 ± 0.21	0.00 ± 0.00	995.8 ± 69.1	τ_{us}
last	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	995.8 ± 69.1	
mean	$\mathbf{0.02 \pm 0.03}$	0.27 ± 0.36	0.01 ± 0.02	992.7 ± 70.7	
first	0.23 ± 0.05	0.18 ± 0.06	0.39 ± 0.12	833.5 ± 118.6	τ_{best}
last	0.17 ± 0.02	0.11 ± 0.02	0.54 ± 0.17	601.3 ± 128.6	
mean	$\mathbf{0.58 \pm 0.08}$	0.69 ± 0.12	0.50 ± 0.09	676.4 ± 103.2	

As has become a pattern, results obtained using the unsupervised version of the PATH dataset and τ_{us} turn out to be of limited use, as the threshold choice veils any differences between the RWMs. Once τ_{best} is used, a clearer picture emerges, however. Even in this unsupervised best-case, both first-type and last-type methods fall short of their mean-type counterpart, in terms of F_1 score, with the first-type method edging out the last-type method. Interestingly, though unexpected, the average detection delay associated with the last-type method is lower than the mean-type method, which, as explained in Chapter 5.2, can be attributed to the higher recall but also fact that any ground-truth anomalies starting after w time steps can be detected with 0 delay.

While all methods show improvements using the semi-supervised version PATH, it does not affect the pecking order observed in the unsupervised version, regardless of what threshold is used. Additionally, the last-type method no longer features the lowest average detection delay, as its theoretical advantage is offset by the higher recall

Table 5.7: Mean ± standard deviation of F_1 score, precision P, recall R, and average detection delay $\bar{\delta}$ using the unsupervised threshold τ_{us} (top half) and hypothetical best threshold τ_{best} (bottom half) for different types of RWM applied to the *semi-supervised* anomaly detection version of the PATH dataset, i.e. the version with anomalous data in training subset $\mathcal{D}^{\text{train}}$. The best F_1 scores are provided in **bold**.

RWM	F_1	P	R	$\bar{\delta}$ $[s]$	
first	0.45 ± 0.15	0.90 ± 0.10	0.31 ± 0.15	934.9 ± 72.5	τ_{us}
last	0.24 ± 0.10	0.84 ± 0.12	0.15 ± 0.07	882.8 ± 77.9	
mean	$\mathbf{0.63 \pm 0.22}$	0.95 ± 0.06	0.52 ± 0.26	589.1 ± 212.0	
first	0.58 ± 0.12	0.66 ± 0.23	0.57 ± 0.14	854.3 ± 81.5	τ_{best}
last	0.40 ± 0.07	0.53 ± 0.29	0.41 ± 0.12	658.8 ± 104.8	
mean	$\mathbf{0.80 \pm 0.15}$	0.88 ± 0.11	0.76 ± 0.18	412.6 ± 143.8	

and therefore lower false negative count of the mean-type method. An RWM with a lower intrinsic delay is therefore of little use, if fewer anomalies can be detected with it.

5.3.4 Generalisation Study

As the last experiment in this section, TeVAE's ability to generalise is further investigated. As specified in Chapter 5.2, the PATH dataset is split into six different folds, with the former three representing cases missing one of three operational ranges, and the latter three representing cases missing one of three levels of dynamics. In this experiment, PATH is split differently to the previous experiments, and hence the results are only loosely comparable. Note that, since the generalisation capability of TeVAE is of interest in this experiment, only τ_{best} is used, as τ_{us} would mask the results obtained. The respective results for both the unsupervised and semi-supervised versions of the PATH dataset are shown in Tables 5.8 and 5.9.

First, consider the results for the first three folds. As is evident, detection performance is best in fold 1 and drops significantly with the other two folds. In other words, the absence of the low average speed drives affects detection performance the least, whereas the absence of the medium and high average speed drives the most, compared to the *All* fold. One possible explanation could be that, all driving cycles start from rest and end at rest, meaning that even those with high average speeds feature some time steps in the lower ranges, therefore covering most of the operational range. In contrast, low average speed cycles only cover a subspace of the operational range, and need to extrapolate at higher speeds, a task that data-driven models are

Table 5.8: Mean ± standard deviation of F_1 score, precision P, recall R, and average detection delay $\bar{\delta}$ using the hypothetical best threshold τ_{best} only for a range of folds applied to the *unsupervised* anomaly detection version of the PATH dataset, i.e. the version with anomalous data in the training subset $\mathcal{D}^{\text{train}}$. Folds one to three represent cases missing sequences of lower, middle, and high average vehicle speeds, respectively, whereas folds four to six represent cases missing sequences of lower, middle, and high levels of dynamics. The so-called *All* fold represents the case where no data is withheld.

Fold	F_1	P	R	$\bar{\delta}$ $[s]$
1	0.51 ± 0.08	0.46 ± 0.08	0.57 ± 0.09	537.0 ± 76.6
2	0.34 ± 0.07	0.27 ± 0.08	0.53 ± 0.15	573.3 ± 110.5
3	0.23 ± 0.02	0.15 ± 0.01	0.52 ± 0.06	567.8 ± 41.0
4	0.49 ± 0.03	0.45 ± 0.00	0.54 ± 0.07	583.4 ± 45.4
5	0.47 ± 0.07	0.42 ± 0.07	0.54 ± 0.08	545.0 ± 71.5
6	0.26 ± 0.01	0.16 ± 0.00	0.64 ± 0.13	484.1 ± 84.2
All	0.55 ± 0.08	0.65 ± 0.11	0.48 ± 0.07	619.8 ± 44.5

particularly weak at. The same pattern can be observed when considering the latter three folds. Clearly, the absence of the most dynamic driving cycles also hurts the detection performance the most, likely for the same reason.

Table 5.9: Mean ± standard deviation of F_1 score, precision P, recall R, and average detection delay $\bar{\delta}$ using the hypothetical best threshold τ_{best} only for a range of folds applied to the *semi-supervised* anomaly detection version of the PATH dataset, i.e. the version with anomalous data in training subset $\mathcal{D}^{\text{train}}$. Folds one to three represent cases missing sequences of lower, middle, and high average vehicle speeds, respectively, whereas folds four to six represent cases missing sequences of lower, medium, and high levels of dynamics. The so-called *All* fold represents the case where no data is withheld.

Fold	F_1	P	R	$\bar{\delta}$ $[s]$
1	0.66 ± 0.13	0.59 ± 0.18	0.79 ± 0.02	344.5 ± 9.1
2	0.49 ± 0.06	0.60 ± 0.29	0.56 ± 0.18	519.0 ± 118.0
3	0.43 ± 0.04	0.99 ± 0.01	0.28 ± 0.03	708.7 ± 22.9
4	0.53 ± 0.14	0.48 ± 0.08	0.61 ± 0.22	502.1 ± 164.8
5	0.64 ± 0.02	0.65 ± 0.11	0.67 ± 0.12	437.3 ± 100.1
6	0.38 ± 0.03	0.63 ± 0.32	0.45 ± 0.26	612.4 ± 180.5
All	0.89 ± 0.01	0.95 ± 0.04	0.84 ± 0.05	318.0 ± 12.3

The semi-supervised results tell much of the same story as the unsupervised results. When considering the first three folds, it is once again clear that the model lacking the higher average speed cycles performs worst. Interestingly, for the latter three folds, it

is the one lacking medium levels of dynamics which performs best, whereas in the case of the unsupervised results, it was the one lacking the low levels of dynamics. Another interesting aspect is that the F_1 scores for folds with the low speeds and low levels of dynamics have a much higher standard deviation.

5.4 Conclusions

Despite Table 5.2 featuring the lowest average detection delay in this entire work, the figure still seems fairly high, as 412.6 s corresponds to almost 7 min delay on average. However, recall that through mean-type RWM a intrinsic delay of $w = 256$ time steps is introduced in almost all cases, corresponding to around 2 min by default. Additionally, recall the nature of the online metrics. They not only punish premature detections with an absolute value of negative delay, but also false negatives with the entire length of the anomalous subsequence within the time series, which further adds to the detection delay.

Overall, the unsupervised and semi-supervised results show that, at least for the best performing approaches, the detection performance relies on both the threshold choice, but also the contamination level in $\mathcal{D}^{\text{train}}$. Therefore, further work needs to be done on truly unsupervised threshold estimation methods or other methods that do not assume labelled data is available for free. Additionally, the robustness of model-based approaches when trained on contaminated training subsets leaves room for improvement and should also be addressed.

Chapter 6

Interactive Threshold Search Using Active Learning

6.1 Introduction

As concluded in Chapter 5.4, there is a need for a methodology which does not assume labelled data is available for free, like is often the case in the literature. To meet this need, active learning can be employed to intelligently choose samples to query labels from a domain expert. The labelled data can then be used to choose a threshold which maximises the F_1 score, similar to the method used to obtain the hypothetical best threshold τ_{best} in Chapter 5.2. The process of choosing samples to be queried is referred to as a query strategy (QS) and is explained in more detail in Chapter 2.6. This chapter is structured as follows. First, Chapter 6.2 provides an overview of other ways active learning is applied to anomaly detection problems. Then, in Chapter 6.3, a novel QS is proposed, named the dissimilarity-based query strategy (DQS), which queries samples most dissimilar to each other, in theory maximising the diversity of labelled sequences. To quantify and compare the detection performance uplifts brought about by DQS, experiments are outlined where it is benchmarked against three other QSs not only using different query budget sizes, but also different mislabelling probabilities. The latter aspect is relevant to the real-world, as even domain experts may make mistakes, though unfortunately the impact of mislabelling has not yet been investigated in literature. In Chapter 6.4 the results of the experiments are then presented and analysed, and finally, in Chapter 6.5, insights gained are once again

recapped and future work is highlighted.

Mathematically, the problem in this chapter can be described as follows. Consider an anomaly detection problem involving some sort of dynamic process running for several days that yields a number of multivariate time series that are sequential w.r.t. each other but are not necessarily temporally contiguous, i.e. a discrete-sequence anomaly detection problem as described in Chapter 2.2. During the dynamic process, each recorded time series $\mathcal{S}$ is added to the candidate set $\mathcal{C}^{\text{ts}} = \{\mathcal{S}_1, ..., \mathcal{S}_n, ..., \mathcal{S}_N\}$ containing all recorded sequences up to the present, where N denotes the number of sequences recorded up to the present and $\mathcal{S}_n \in \mathbb{R}^{T_n \times d_{\mathcal{D}}}$, where T_n denotes the number of time steps in $\mathcal{S}_n$, and $d_{\mathcal{D}}$ the dimensionality of the data set. Now, consider an anomaly detector $\mathcal{M}$ that has been used to check the data for anomalous behaviour and yields a univariate anomaly score $\mathbf{s}_n = \mathcal{M}(\mathcal{S}_n)$, where $\mathbf{s}_n \in \mathbb{R}^{T_n}$. The anomaly score counterpart to the candidate set $\mathcal{C}^{\text{ts}}$ is denoted as the candidate *score* set $\mathcal{C}^{\text{as}}$ and is structured similarly, such that $\mathcal{C}^{\text{as}} = \{\mathbf{s}_1, ..., \mathbf{s}_n, ..., \mathbf{s}_N\}$.

In the context of active learning with time series data, a sequence is referred to as a sample, where a subset of sequences of $\mathcal{C}^{\text{ts}}$ is chosen to be sent to the oracle, i.e. the domain expert. Once the query budget B is used up, i.e. B time series are chosen, they are moved to a set called the query set $\mathcal{Q}^{\text{ts}}$ which contains all time series selected for querying up to the present. The anomaly score counterpart to the query set $\mathcal{Q}^{\text{ts}}$ is denoted as the query score set $\mathcal{Q}^{\text{as}}$ and is structured similarly. Once the query score set is established, it can be used to improve the anomaly detection performance using the feedback strategy (FS).

6.2 Related Work

The wide majority of model-based approaches found in the literature use some sort of anomaly score, which indicates how anomalous each time step within a test subset sequence is. To classify time steps in the anomaly score as nominal or anomalous, a threshold needs to be set, however, many publications in unsupervised anomaly detection circumvent the issue by using labelled data, technically making the respective approaches supervised. The labelled data is either used to set a threshold which satisfies a condition, like achieving the maximum possible F_1 score or some other parametric threshold setting method is proposed, like the peaks over threshold method [87], which has found wide application throughout the literature [77, 88–90]. Hundman et al. [32] introduce the non-parametric dynamic threshold, but unfortunately, despite *non-parametric* being in the name, it is not truly non-parametric, as the threshold

can only be set by setting a hyperparameter **z** which is experimentally set, and therefore requires labelled data. In addition to that, this method cannot be applied to discrete-sequence anomaly detection, as it assumes a predicted anomalous time steps or ground-truth anomalous time steps within a given test sequence. When neither is given, like in the case of a correctly identified nominal time series, the threshold becomes undefined.

Generally, active learning is rarely used to adjust thresholds in model-based time series anomaly detection approaches. Literature dealing with further training the anomaly detection model exists, but such methods are not generically applicable, since they modify loss functions to penalise wrong classifications [105–108] or utilise an architecture specific to active learning which also requires labelled data from the beginning [109].

Lundström et al. [110], on the other hand, propose a threshold setting method for continuous-sequence problems. First, it initialises a threshold based on a variation of the "non-parametric" dynamic threshold [32], then, anomalous sub-sequences are clustered according to four pre-defined attributes of temporally adjacent sub-sequences. According to Lundström et al. [110] the mathematical definition of *temporally adjacent* implies that two anomalous sub-sequences can be temporally adjacent if they occur after one another, regardless of whether there is another sub-sequence between them, though in this work their explanation is interpreted as referring to two sub-sequences following one another *directly*, i.e. there is no other anomalous sub-sequence between them. The first attribute, A_1 denotes the temporal distance between the last time step of a predicted anomalous sub-sequence and the first time step of the following predicted anomalous sub-sequence. The second and third attributes, A_2 and A_3, denote the absolute difference between the respective maximums of two temporally adjacent predicted anomalous sub-sequences. The difference between the second and third attributes is that the former looks for the maximum value in each channel of the input time series, whereas the latter looks for the maximum value in each channel of the anomaly score. The fourth attribute A_4 is a binary vector, which assumes a 1 if both maximum anomaly scores in each channel of two temporally adjacent subsequences exceed the current threshold. As is evident, all but the first attribute are obtained feature-wise, i.e. $A_2, A_3, A_4 \in \mathbb{R}^{d_{\mathcal{D}}}$. Following the calculation of attributes, Lundström et al. [110] aggregate them into several logical outputs, which assume a binary value depending on whether the respective attribute exceeds the set threshold for that attribute. The logical outputs are then combined to decide whether two temporally adjacent predicted anomalous sub-sequences should be clustered together. Interestingly,

this threshold setting method requires pre-defined thresholds to work, and therefore it needs to be parametrised. Additionally, it cannot be applied to discrete-sequence problems since, to obtain attribute A_1, predicted anomalous sub-sequences need to be within the same sequence, which may not always be the case.

There is a clear need for methods which allow for the integration of active learning into the threshold choice for discrete-sequence anomaly detection problems. Additionally, all the above-mentioned literature assumes perfect labelling, however, as pointed out by Settles [111], labels from human experts are not always reliable as some samples are difficult to label and humans can be distracted or fatigued over time. Therefore, this work investigates the impact of different rates of mislabelling on the anomaly detection performance, which has not been undertaken previously in the literature.

6.3 Proposed Methodology

6.3.1 Dissimilarity-based Querying Strategy

Generally, query strategies work by letting an oracle label B times each round, i.e. the number of samples allowed to be queried in a given round q. While it is possible to choose B sequences randomly every round, there is little control over the diversity of the queried sequences. In this work, the diversity of sequences is maximised by actively choosing samples most dissimilar to the previously queried ones, by inspecting the dynamic time warping (DTW) distance between the samples in question, i.e. the anomaly scores of different time series, where a high distance is interpreted as poor alignment of features within the anomaly scores and, therefore, a high level of dissimilarity. For a more detailed explanation of DTW, please refer to Chapter 2.7. Unlike other metrics, like the mean-squared-error, dynamic time-warping distance does not require two sequences to be the same length, which makes it ideal for an application with variable-length sequences. Compared to univariate time series, calculating the dissimilarity between multivariate time series in the same way is possible, however, it is much more computationally intensive.

The exact procedure for DQS is as follows and is also depicted as pseudocode in Algorithm 6. For the first query in the first round, i.e. $b = q = 1$ and $|\mathcal{Q}^{\text{as}}| = 0$, DQS is initialised by choosing a random anomaly score with index r from the candidate score set $\mathcal{C}^{\text{as}}$, as shown in line 3 in Algorithm 6. It is then moved to the query score set $\mathcal{Q}^{\text{as}}$, such that $|\mathcal{Q}^{\text{as}}| = 1$, as shown in line 16. Once the querying is initialised, for the next query, i.e. $b = 2$, the anomaly score in $\mathcal{C}^{\text{as}}$ most similar to any anomaly score

Algorithm 6 DQS at an arbitrary round q

Require: Candidate score set $\mathcal{C}^{\text{as}}$, Query score set $\mathcal{Q}^{\text{as}}$, Budget B

1: **for** $b = 1 \rightarrow B$ **do**
2: **if** $|\mathcal{Q}^{\text{as}}| = 0$ **then**
3: $r \sim \mathcal{U}(1, |\mathcal{C}^{\text{as}}|)$ ▷ Random index for initialisation
4: **else**
5: **for** $n = 1 \rightarrow |\mathcal{C}^{\text{as}}|$ **do**
6: **for** $m = 1 \rightarrow |\mathcal{Q}^{\text{as}}|$ **do**
7: $\mathbf{D}[n, m] = DTW(\mathcal{C}^{\text{as}}[n], \mathcal{Q}^{\text{as}}[m])$
8: **end for**
9: **end for**
10: $o = \arg\min(\mathbf{D})$ ▷ Index most similar to any sample in $\mathcal{Q}^{\text{as}}$
11: **for** $n = 1 \rightarrow |\mathcal{C}^{\text{as}}|$ **do**
12: $\mathbf{E}[n] = DTW(\mathcal{C}^{\text{as}}[n], \mathcal{C}^{\text{as}}[o])$
13: **end for**
14: $r = \arg\max(\mathbf{E})$ ▷ Index most dissimilar to $\mathcal{C}^{\text{as}}[o]$
15: **end if**
16: $\mathcal{Q}^{\text{as}} \leftarrow \mathcal{C}^{\text{as}}[r]$ ▷ Move $\mathcal{C}^{\text{as}}[r]$ to $\mathcal{Q}^{\text{as}}$
17: **end for**
18: **return** Query Score Set $\mathcal{Q}^{\text{as}}$

within $\mathcal{Q}^{\text{as}}$ is tagged; its index is denoted as o, as shown in lines 5 to 10. Following that, the anomaly score in $\mathcal{C}^{\text{as}}$ most dissimilar to the tagged $\mathcal{C}^{\text{as}}[o]$, denoted by index r, is moved to $\mathcal{Q}^{\text{as}}$, such that $|\mathcal{Q}^{\text{as}}| = 2$, as shown in lines 11 to 14. This is repeated until B sequences are moved to $\mathcal{Q}^{\text{as}}$, such that, at the end of round 1, $|\mathcal{Q}^{\text{as}}| = B$. For round 2, no initialisation is required, and hence the procedure is the same as round 1 for $b > 1$. Therefore, after round 2, B anomaly scores are moved to the query score set $|\mathcal{Q}^{\text{as}}| = 2 \cdot B$. The following rounds after that follow the same pattern. Clearly, the query set and the candidate set both grow with every round, complicating computation as time goes on.

Once a round is over and B new samples are added to the query score set $\mathcal{Q}^{\text{as}}$, the threshold τ is grid-searched to maximise the F_1 score based on the labels provided by the oracle, though future work could investigate the feasibility of more efficient search methods.

This method is generically applicable to discrete-sequence problems that rely on a threshold as a decision boundary between nominal and anomalous sequences.

Obviously, a QS can only be employed in cases where a domain expert is available. Cases where this does not apply will benefit from more sophisticated ways to choose a truly unsupervised threshold.

6.3.2 Experimental Setup

Competing Approaches

As mentioned in Chapter 6.1, DQS is benchmarked against three other QS. The first, named the random-based query strategy (RQS), represents a scenario where, rather than intelligently querying an oracle, the samples to be labelled are chosen at random, as shown in Algorithm 7.

Algorithm 7 RQS at an arbitrary round q

Require: Candidate Score Set $\mathcal{C}^{\text{as}}$, Query Score Set $\mathcal{Q}^{\text{as}}$

for $b = 1 \rightarrow B$ **do**
  $r \sim \mathcal{U}(1, |\mathcal{C}^{\text{as}}|)$ ▷ Random index is sampled
  $\mathcal{Q}^{\text{as}} \leftarrow \mathcal{C}^{\text{as}}[r]$
end for
return Query Score Set $\mathcal{Q}^{\text{as}}$

The second, named the top-based query strategy (TQS) [105–107, 109, 112, 113], represents a scenario where the B samples with the highest anomaly score are queried, as shown in Algorithm 8. Some literature [27, 114, 115] also refers to this QS as uncertainty-based, as they interpret a high anomaly score as high uncertainty.

Algorithm 8 TQS at an arbitrary round q

Require: Candidate Score Set $\mathcal{C}^{\text{as}}$, Query Score Set $\mathcal{Q}^{\text{as}}$

for $b = 1 \rightarrow B$ **do**
  $r = \arg\max(\mathcal{C}^{\text{as}})$ ▷ Index of maximum anomaly score
  $\mathcal{Q}^{\text{as}} \leftarrow \mathcal{C}^{\text{as}}[r]$
end for
return Query Score Set $\mathcal{Q}^{\text{as}}$

The third, named the uncertainty-based query strategy (UQS), represents a scenario where the B samples closest to the threshold are queried, as shown in Algorithm 9. This strategy needs to be initialised with a threshold, however, in literature, this is not always defined [107, 112, 113] or it relies on labelled data [109].

Benchmarking Considerations

This work performs all experiments on the powertrain anomaly time series benchmark (PATH) data set, a discrete-sequence data set that provides a large number of diverse of multivariate time series and non-trivial anomalies. The dataset represents the dynamic behaviour of an automotive powertrain at different states, which is generated using

Algorithm 9 UQS at an arbitrary round q

Require: Candidate Score Set $\mathcal{C}^{\text{as}}$, Query Score Set $\mathcal{Q}^{\text{as}}$
if $|\mathcal{Q}^{\text{as}}| = 0$ **then**
  $\bar{\mathcal{C}}_n^{\text{as}} = \frac{1}{T_n} \sum_{t=1}^{T_n} \mathbf{s}_n$
  $\tau = \frac{1}{N} \sum_{n=1}^{N} (\bar{\mathcal{C}}_n^{\text{as}})$
end if
for $b = 1 \rightarrow B$ **do**
  $r = \arg\min(\text{abs}(\mathcal{C}^{\text{as}} - \tau))$ ▷ Index of nearest anomaly score to τ
  $\mathcal{Q}^{\text{as}} \leftarrow \mathcal{C}^{\text{as}}[r]$
end for
return Query Score Set $\mathcal{Q}^{\text{as}}$

state-of-the-art simulation tools to exhibit real-world-like complexity, as described in Chapter 4.1. Other publicly available datasets exist, though none are of discrete-sequence nature and this size and diversity, as shown in Chapter 3.1. To adapt the PATH data set for active learning, the unlabelled data subset is split such that the time series within it represent a day's worth of measurements, two days worth of measurements, etc. until the entire subset is depleted, which results in 34 increasingly larger splits. Therefore, for the day 1 split, the sum of the durations of each sequence within the split is roughly equal to 24 hours, for the day 2 split the sum is roughly equal to 48 hours, and so forth. This mimics the gradual collection of data with time, like in the real world. To maintain reasonable computational complexity, it is not feasible to perform active learning every day, therefore a querying round of B samples is done after day 1, day 7, day 14, day 21, and day 28. On each of the days considered, a new model is also trained, such that the new threshold obtained using active learning is applied to the newly trained model. Like in Chapter 5.2, the validation subset size is set as 20 % of the unlabelled subset.

To provide context to the results obtained for the different QS, two baselines will also be provided to the results. The first represents a scenario where the unsupervised threshold τ_{us} is used after each model training, whereas the second represents a hypothetical scenario where the best threshold τ_{best} is known. It is obtained by grid-searching through the test subset and choosing the threshold yielding the highest F_1 score. Neither of the scenarios involve active learning, i.e. no samples are sent to the oracle for labelling.

Like in Chapter 5.3, the online evaluation metrics are obtained for each fold, though unlike in that chapter, the modelling process is not the focus. The query and feedback strategies occur *after* model training and highly depend on the seed used due to

the random operations in the initialisation of the query strategies and mislabelling mechanism; therefore, the results for each fold are collected obtained using seeds 1 to 3, after which the average for each evaluation metric is provided.

The active learning-based approaches tested in this work always use the most recent model available and assume that the oracle is queried once every day at the end of the day. For each split, labels for queried sequences in earlier splits are also considered.

Environment

Temporal VAE (TeVAE) is used as the underlying model and is trained as specified in Chapter 5.2, except with varying training subset sizes. The framework used for model training is TensorFlow 2.15.1 and TensorFlow Probability 0.23 on Python 3.10 on a workstation running Ubuntu 22.04.5 LTS, equipped with two Nvidia RTX A6000 GPUs. Further information on library versions used can be found in the *requirements.txt* file in the GitHub repository under `github.com/lcs-crr/Thesis`.

6.4 Results

6.4.1 Baselines without Active Learning

First, a selection of baseline results is presented in Table 6.1. The table shows the results obtained using the unsupervised threshold τ_{us}, i.e. the threshold obtained by taking the maximum value anomaly score observed in the validation subset, and the results obtained hypothetical best threshold τ_{best}, i.e. the threshold that maximises the F_1 score. The latter is referred to as *hypothetical* since it is obtained using the test subset, which in the real-world is not available, making this threshold unobservable outside this experimental setting.

Table 6.1: F_1 scores for each of the splits using the unsupervised threshold τ_{us} and the hypothetical best threshold τ_{best}.

Baseline	1 Day	7 Days	14 Days	21 Days	28 Days
Unsupervised	0.16 ± 0.03	0.16 ± 0.01	0.12 ± 0.10	0.06 ± 0.04	0.04 ± 0.02
Best	0.20 ± 0.05	0.38 ± 0.09	0.45 ± 0.10	0.58 ± 0.13	0.68 ± 0.10

As expected, the results using the hypothetical best threshold τ_{best} are higher across the board. There is a downward trend in the F_1 score obtained with the unsupervised threshold τ_{us} as time goes on and the validation subset grows. This is because, with a

larger validation subset and therefore a higher anomaly count within it, the probability of a very high anomaly score grows, leading to a higher-than-ideal threshold setting. *The goal of any QS is to outperform the unsupervised threshold τ_{us}, while coming as close to the hypothetical best as possible.*

6.4.2 Impact of Query Budgets

A large query budget B is associated with a higher labelling cost, and hence it is desirable to use a QS that maximises the F_1 score improvement with a limited number of queries. To investigate the impact the query budget has on the anomaly detection results, three different budget sizes are used: $B = 1$, $B = 5$, and $B = 10$. In this experiment, it is assumed that all queries are correctly labelled.

The results for different query strategies (random-based, top-based, uncertainty-based, dissimilarity-based) and different query budgets ($B = 1$, $B = 5$, $B = 10$) are presented in Table 6.2 and, to further aid interpretation, as a function of time in Figure 6.1.

Table 6.2: F_1 scores for each of the splits for different query budgets using different query strategies and the DQS. The results obtained using the unsupervised threshold τ_{us} and the hypothetical best threshold τ_{best} are also presented for reference. The best results for the respective split at a given budget size B is shown in **bold**. The values shown consist of the mean ± standard deviation over the three folds, each on the three different seeds. A plot of the F_1 score as a function of time is shown in Figure 6.1.

	B	1 Day	7 Days	14 Days	21 Days	28 Days
τ_{us}	-	0.16 ± 0.03	0.16 ± 0.01	0.12 ± 0.10	0.06 ± 0.04	0.04 ± 0.02
τ_{best}	-	0.20 ± 0.05	0.38 ± 0.09	0.45 ± 0.10	0.58 ± 0.13	0.68 ± 0.10
RQS	1	$\mathbf{0.10 \pm 0.01}$	0.10 ± 0.01	0.10 ± 0.01	0.10 ± 0.01	0.16 ± 0.16
TQS	1	$\mathbf{0.10 \pm 0.01}$	0.19 ± 0.13	0.14 ± 0.05	0.21 ± 0.14	0.24 ± 0.17
UQS	1	$\mathbf{0.10 \pm 0.01}$	0.10 ± 0.01	0.10 ± 0.01	0.13 ± 0.04	0.14 ± 0.06
DQS	1	$\mathbf{0.10 \pm 0.01}$	$\mathbf{0.28 \pm 0.11}$	$\mathbf{0.31 \pm 0.14}$	$\mathbf{0.39 \pm 0.18}$	$\mathbf{0.53 \pm 0.19}$
RQS	5	0.11 ± 0.02	0.16 ± 0.06	0.29 ± 0.18	0.39 ± 0.23	0.50 ± 0.25
TQS	5	0.14 ± 0.06	$\mathbf{0.28 \pm 0.06}$	0.29 ± 0.02	0.30 ± 0.06	0.26 ± 0.12
UQS	5	0.14 ± 0.06	0.17 ± 0.11	0.17 ± 0.11	0.20 ± 0.15	0.20 ± 0.14
DQS	5	$\mathbf{0.15 \pm 0.05}$	0.27 ± 0.07	$\mathbf{0.31 \pm 0.15}$	$\mathbf{0.43 \pm 0.19}$	$\mathbf{0.61 \pm 0.13}$
RQS	10	0.12 ± 0.03	0.18 ± 0.06	0.27 ± 0.17	0.44 ± 0.22	$\mathbf{0.60 \pm 0.15}$
TQS	10	$\mathbf{0.16 \pm 0.05}$	$\mathbf{0.28 \pm 0.07}$	0.31 ± 0.12	0.44 ± 0.06	0.51 ± 0.14
UQS	10	$\mathbf{0.16 \pm 0.05}$	0.18 ± 0.07	0.22 ± 0.14	0.27 ± 0.12	0.35 ± 0.18
DQS	10	$\mathbf{0.16 \pm 0.05}$	$\mathbf{0.28 \pm 0.06}$	$\mathbf{0.39 \pm 0.10}$	$\mathbf{0.50 \pm 0.12}$	0.46 ± 0.23

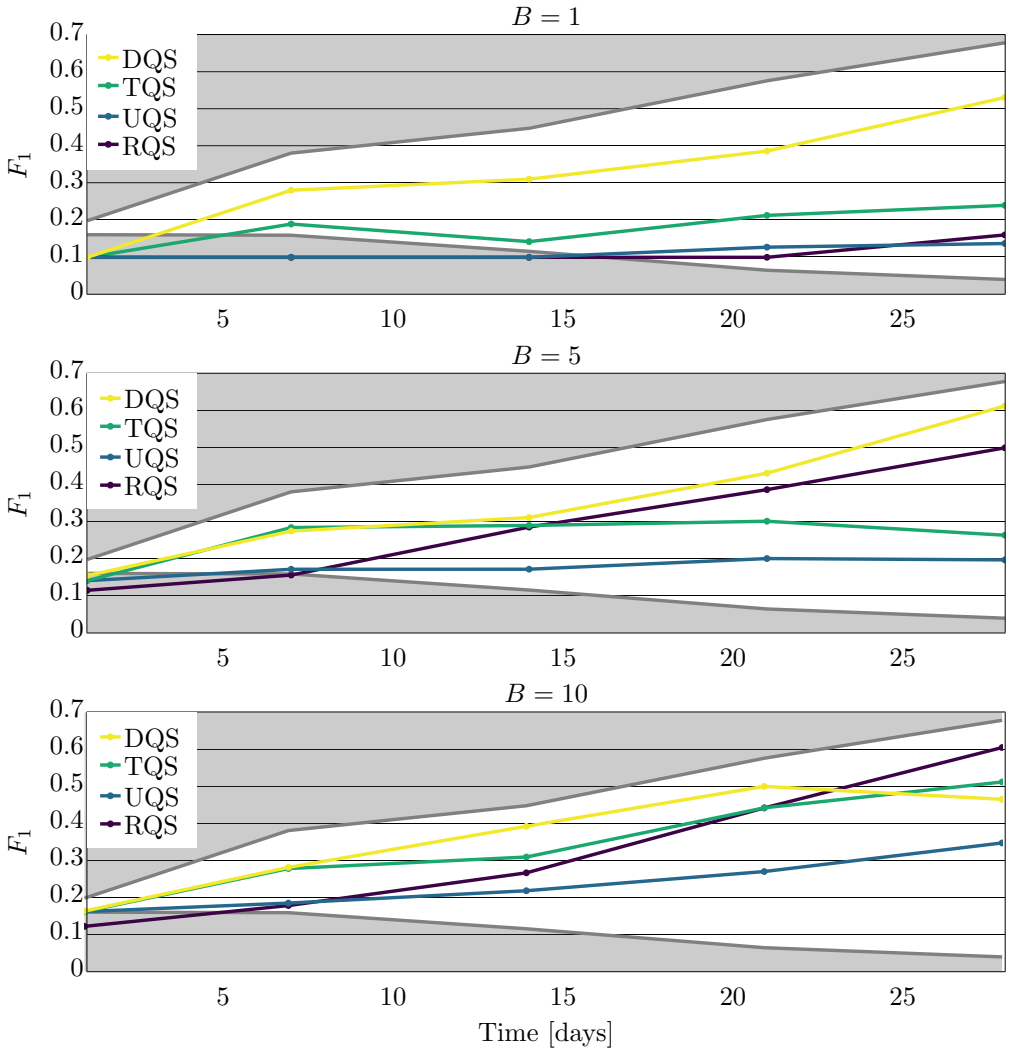

Figure 6.1: F_1 scores for different query budgets at a fixed mislabelling probability of $p_m = 0$ plotted as a function of time. The results in tabular form are also shown in Table 6.2. The grey upper bound represents the results using the hypothetical best threshold τ_{best}, and the grey lower bound represents the results using the unsupervised threshold τ_{us}.

The first observation that can be made from Table 6.2 and Figure 6.1 is that, at the smallest split of 1 day for all budget sizes, all query strategies perform worse or, at most, comparably to the scenario using the unsupervised threshold τ_{us}. This can be attributed to the poor model quality at the small training subset size, indicated by the low hypothetical best results. *As the training subset grows, the difference between*

the two baselines grows, which is where active learning can have the most impact. At small budget sizes, i.e. at $B < 10$, the DQS outperforms the other query strategies across the board. At $B = 10$, the DQS outperforms or is at most matched by the other query strategies except for the last split of 28 days, where the RQS performs the best. Lastly, at most budget sizes and splits, the UQS falls short of the other query strategies.

6.4.3 Impact of Mislabelling Probability

While it is assumed the oracle in this work is a domain expert, they can still make mistakes. In literature, the effect of mislabelling has not been investigated as far the author is aware, therefore, for a fixed budget size of $B = 10$ queries, a mislabelling probability of $p_m = 0.1$, $p_m = 0.2$, $p_m = 0.3$ is experimented with. This is done by sampling from a Bernoulli distribution parameterised with the above-mentioned mislabelling probabilities and flipping ground-truth labels accordingly. The choice of the Bernoulli distribution is based on several reasons. For one, unlike a Binomial distribution, which is a function of the number of trials, the Bernoulli distribution is of single-trial nature. Additionally, the Bernoulli distribution is discrete and binary, sampling it yields a boolean, corresponding to mislabelled or correctly labelled.

The results for different query strategies (random-based, top-based, uncertainty-based, dissimilarity-based) and different mislabelling probabilities ($p_m = 0.1$, $p_m = 0.2$, $p_m = 0.3$) are presented in Table 6.3 and, to further aid interpretation, as a function of time in Figure 6.2.

Unsurprisingly, the above-made observations regarding the low F_1 scores at the smallest split, can also be made when investigating the impact of mislabelling on detection performance. Aside from the results at the smallest split, all query strategies perform comparably at $p_m = 0.1$, with DQS performing slightly better. This holds mostly true for $p_m = 0.2$ and $p_m = 0.3$ as there is no clear winner for all split sizes. What is clear, is that at the two higher mislabelling probabilities, the RQS performs worst. Overall, the results scatter more between query strategies as the mislabelling probability increases. Interestingly, some results improve despite higher mislabelling probabilities. This can happen when the mislabelled samples do not change the position of the threshold, for example, when an equal number of samples are mislabelled on either side of the threshold.

Table 6.3: F_1 scores for each of the splits for a query budget of $B = 10$ and different mislabelling probabilities using different query strategies and the DQS. The results obtained using the unsupervised threshold τ_{us} and the hypothetical best threshold τ_{best} are also presented for reference. The best results for the respective split at a given mislabelling probability p_m is shown in **bold**. The values shown consist of the mean ± standard deviation over the three folds, each on the three different seeds. A plot of the F_1 score as a function of time is shown in Figure 6.2.

	p_m	1 Day	7 Days	14 Days	21 Days	28 Days
τ_{us}	-	0.16 ± 0.03	0.16 ± 0.01	0.12 ± 0.10	0.06 ± 0.04	0.04 ± 0.02
τ_{best}	-	0.20 ± 0.05	0.38 ± 0.09	0.45 ± 0.10	0.58 ± 0.13	0.68 ± 0.10
RQS	0	0.12 ± 0.03	0.18 ± 0.06	0.27 ± 0.17	0.44 ± 0.22	$\mathbf{0.60 \pm 0.15}$
TQS	0	$\mathbf{0.16 \pm 0.05}$	$\mathbf{0.28 \pm 0.07}$	0.31 ± 0.12	0.44 ± 0.06	0.51 ± 0.14
UQS	0	$\mathbf{0.16 \pm 0.05}$	0.18 ± 0.07	0.22 ± 0.14	0.27 ± 0.12	0.35 ± 0.18
DQS	0	$\mathbf{0.16 \pm 0.05}$	$\mathbf{0.28 \pm 0.06}$	$\mathbf{0.39 \pm 0.10}$	$\mathbf{0.50 \pm 0.12}$	0.46 ± 0.23
RQS	0.1	0.11 ± 0.02	0.21 ± 0.06	0.25 ± 0.15	0.38 ± 0.17	0.53 ± 0.21
TQS	0.1	$\mathbf{0.16 \pm 0.05}$	0.27 ± 0.07	0.32 ± 0.06	0.44 ± 0.19	0.50 ± 0.13
UQS	0.1	$\mathbf{0.16 \pm 0.05}$	0.19 ± 0.09	0.30 ± 0.12	0.43 ± 0.22	0.43 ± 0.11
DQS	0.1	0.13 ± 0.05	$\mathbf{0.29 \pm 0.06}$	$\mathbf{0.36 \pm 0.13}$	$\mathbf{0.50 \pm 0.15}$	$\mathbf{0.57 \pm 0.17}$
RQS	0.2	0.12 ± 0.02	0.17 ± 0.06	0.20 ± 0.14	0.20 ± 0.10	0.43 ± 0.17
TQS	0.2	$\mathbf{0.16 \pm 0.05}$	$\mathbf{0.33 \pm 0.08}$	$\mathbf{0.33 \pm 0.06}$	0.43 ± 0.05	0.51 ± 0.15
UQS	0.2	$\mathbf{0.16 \pm 0.05}$	0.23 ± 0.11	0.29 ± 0.12	0.44 ± 0.24	0.46 ± 0.16
DQS	0.2	0.11 ± 0.04	0.23 ± 0.10	$\mathbf{0.33 \pm 0.13}$	$\mathbf{0.46 \pm 0.17}$	$\mathbf{0.54 \pm 0.17}$
RQS	0.3	0.11 ± 0.02	0.15 ± 0.06	0.20 ± 0.16	0.15 ± 0.05	0.21 ± 0.12
TQS	0.3	$\mathbf{0.14 \pm 0.06}$	$\mathbf{0.35 \pm 0.06}$	0.28 ± 0.12	$\mathbf{0.52 \pm 0.15}$	0.47 ± 0.14
UQS	0.3	0.14 ± 0.06	0.26 ± 0.12	$\mathbf{0.31 \pm 0.12}$	0.37 ± 0.26	0.40 ± 0.15
DQS	0.3	0.11 ± 0.04	0.23 ± 0.11	0.30 ± 0.17	0.33 ± 0.21	$\mathbf{0.55 \pm 0.15}$

6.5 Conclusions

The results show that, there is no QS that outperforms all others in every category. While DQS performs best in small-budget cases and remains competitive at higher ones, TQS demonstrates higher robustness when faced with mislabelling. In the real world, it therefore depends on the expertise of the oracle and how many samples they are willing to label. In any case, except for very small splits, all querying strategies outperform the unsupervised threshold τ_{us} despite the presence of mislabelling, and hence, based on the experiments undertaken on the PATH dataset, it is advisable to use an active learning-based threshold if the possibility to query an oracle exists.

Future work should focus on standardising the benchmarking procedure of active learning methods in time series anomaly detection and should include investigation

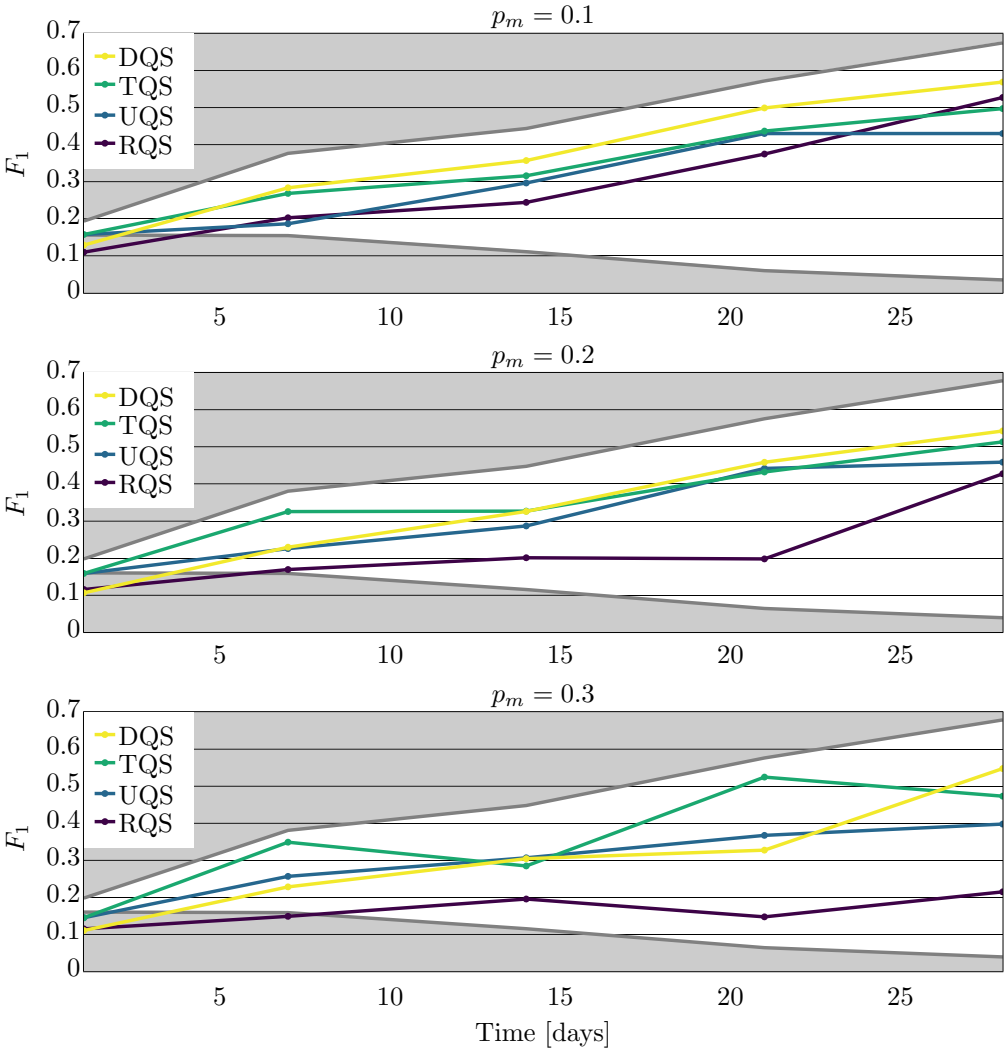

Figure 6.2: F_1 scores for different mislabelling probabilities at a fixed query budget of $B = 10$ plotted as a function of time. The results in tabular form are also shown in Table 6.3. The grey upper bound represents the results using the hypothetical best threshold τ_{best}, and the grey lower bound represents the results using the unsupervised threshold τ_{us}.

on the impact of mislabelling, proposed in this work. Furthermore, creating a QS that not only performs well with small query budgets but is also robust in the case of mislabelling should be further researched. Perhaps this can be achieved by measuring dissimilarity differently, either by using of a different metric besides DTW or by comparing other aspects, like the reconstructions in the case of autoencoders.

Chapter 7

Extending Representation Learning to Predictive Maintenance

7.1 Introduction

As demonstrated in Chapter 5.3, modelling nominal behaviour of dynamic systems allows for the detection of anomalies when no examples of anomalies are present. This procedure can also useful for other time series problems, such as predictive maintenance (PdM), which contrasts simpler maintenance strategies like run to failure (RTF) and preventative maintenance (PM). On the one hand, RTF involves running arbitrary dynamic systems until failure, and therefore leads to unexpected downtime, as well as higher costs and potential damage to other, otherwise healthy, components. On the other hand, PM involves the preemptive change of components in said arbitrary system before they break, also leading to higher costs, as components are changed despite having remaining useful life. PdM involves predicting the state of health of the dynamic system, which can then be used to inform a maintainer on the ideal time for maintenance, i.e. the time before a component within the system breaks but also has little remaining useful life.

Unlike anomaly detection, PdM does not try to separate data into two classes, as component deterioration is usually a gradual process. Modelling the nominal behaviour at the beginning of life of a dynamic system therefore allows for the quantifi-

cation of the deterioration level by comparing the model behaviour with the observed behaviour, since a real-world system experiences wear, whereas the model does not.

To illustrate the applicability of a data-driven approach for time series anomaly detection, like the temporal VAE (TeVAE), it is investigated based on a real-world case study. This real-world problem involves an electric motor (EM) on a test bench, with which a single drive cycle is driven repeatedly. Similar to the powertrain introduced in Chapter 4.2, the EM is connected to two resistances which simulate the real-world road load. This EM consists of a stator and a rotor, which are both cooled and lubricated by an oil-filled loop. Due to the rotating nature of the rotor, a special manifold is used to distribute the oil, which, in this case study, twists with time due to human error during installation. The twisting motion is very gradual but leads to a reduction of the cross-sectional area the oil can flow through, impacting the cooling capacity available to the rotor. This would usually manifest itself in features like the rotor temperature, though, given the rotating nature, no sensors are present to quantify such properties. Oil pressure, in contrast, is monitored by a simple rule-based system, and it could be used to detect the reduction of cross-sectional area, though this is a quantity that would not typically be relevant unless this type of failure is anticipated. Considering this, as well as the absence of a physical metric that can serve as a health indicator, and the lack of labelled failure data due to twisted oil distributors, this is classified as an unsupervised PdM problem [116, 117], which can be approached by computing a virtual health indicator in form of the anomaly score output by TeVAE. Unlike a physical health indicator, though, a virtual health indicator can no longer be interpreted as the quantification of a physical condition.

Additionally, this EM has an integrated two-stage gearbox, which decouples its angular speed from the axle angular speed. Earlier chapters in this thesis did not have to consider such a change in discrete states, hence TeVAE is modified to regard the current gear signal as an exogenous input feature which is not reconstructed and therefore does not contribute to the health indicator.

7.2 Experimental Setup

The features selected to represent testee behaviour are once again based on domain knowledge, and somewhat follow the choices made in Chapter 4. They are shown in tabulated form in Table 7.1. Additionally, Figure 7.1 shows an example of a test sequence plotted for each of the channels.

The available dataset $\mathcal{D}$ consists of 1048 tests, which after an applied training and

Table 7.1: Channels (features) chosen for modelling testee behaviour.

Index	Signal Name	Index	Signal Name
1	Vehicle Speed	7	Inverter Temperature
2	EM Voltage	8	Left Axle Torque
3	EM Current	9	Right Axle Torque
4	EM Torque	10	Left Axle Angular Speed
5	EM Angular Speed	11	Right Axle Angular Speed
6	EM Stator Temperature	12	Current Gear

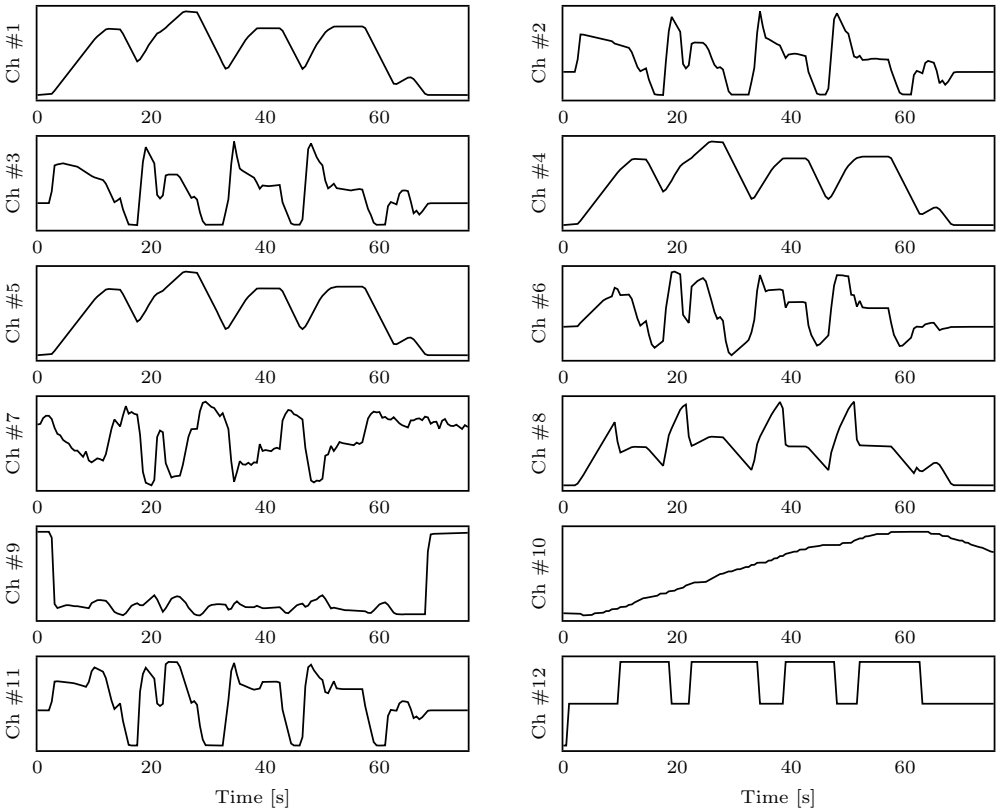

Figure 7.1: Sample plot of a test sequence. The channel legend can be found in Table 7.1. The y-axis ticks are removed due to comply with confidentiality requirements.

test split of 1:2 corresponds to a training subset $\mathcal{D}^{\text{train}}$ of size $M = 349$ and a test subset $\mathcal{D}^{\text{test}}$ of size $N = 699$. It should be noted that, unlike previous experiments, $\mathcal{D}$ is not shuffled before being split to maintain the temporal order of tests. Like in Chapter 5.2, 20 % of the unlabelled training subset $\mathcal{D}^{\text{train}}$ is separated to form the validation subset

$\mathcal{D}^{\text{val}}$, later used for early stopping and threshold estimation. Following the procedure from Chapter 4, the data is low-pass-filtered and downsampled to a sampling frequency of 2 Hz and then windowed to a window size of $w = 16$ time steps. As is evident, the window size is much smaller compared to the datasets discussed in Chapter 4, which stems from the fact that, unlike the others, this dynamic system lacks a battery with the relatively slow temperature and state of charge signals.

TeVAE is used as the underlying model and is trained as specified in Chapter 5.2. The framework used for model training is TensorFlow 2.15.1 and TensorFlow Probability 0.23 on Python 3.10 on a workstation running Ubuntu 22.04.5 LTS, equipped with two Nvidia RTX A6000 GPUs. Further information on library versions used can be found in the *requirements.txt* file in the GitHub repository under `github.com/lcs-crr/Thesis`.

7.3 Results

When TeVAE is applied to the test subset $\mathcal{D}^{\text{test}}$, a health indicator as a function of time for each sequence is generated. To illustrate the development of the health indicator with time, the maximum value of each sequence is taken and plotted against the test sequence index in Figure 7.2.

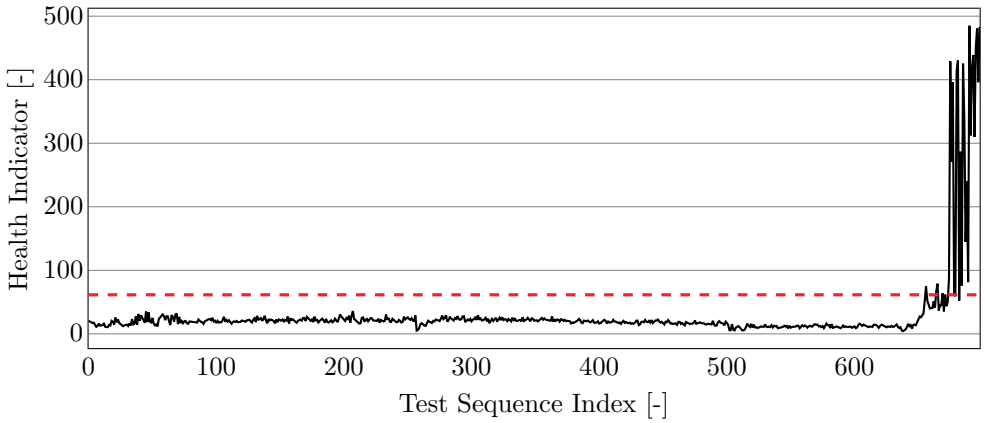

Figure 7.2: Health indicator plotted as a function of the test sequence index. The red dashed line represents the unsupervised threshold obtained using the health indices within the validation subset.

As is evident, the health indicator remains largely constant for most of the test set. At test sequence index 656, the health indicator first crosses the threshold, which

serves as an indicator for users that the model behaviour is significantly deviating from the observed behaviour and that the system or a component within it is starting to deteriorate. The rule-based oil pressure monitor only warned the user at the last test sequence and, considering that each test sequence is around 75 s long, this corresponds to an improvement in deterioration detection by around 54 min.

Considering that TeVAE is essentially used in the same way as it was in Chapter 5.2, it can serve as an approach for both time series anomaly detection and predictive maintenance. During inference, once a sequence is input and reconstructed by TeVAE, it can be used for both problems at no additional computational cost, setting a low hurdle for such a dual operation in the real world.

7.4 Conclusions

This chapter shows that, in the case of a real-world predictive maintenance problem, TeVAE can quantify the change in system behaviour due to deterioration. This is especially remarkable considering that the two-fold unsupervised nature of the problem, which encompasses the lack of a physical and observable health indicator, as well as the absence of labelled historical data depicting failure due to a twisted oil distributor.

Future work should revolve around providing a more sophisticated, perhaps component-level, health indicator, as currently the health indicator only represents the health of the system as a whole. A component-level health indicator allows for more precise diagnostics of a potential problem, especially with very complex systems consisting of several subsystems.

Chapter 8

Conclusions and Outlook

This thesis represents a summary of the research done in the field of time series anomaly detection. As identified in Chapter 3, several gaps in the literature are identified and formulated in the form of research questions in Chapter 1.2. This thesis addresses these by contributing to the aspect of benchmarking, robust modelling, as well as intelligent querying of labels to inform a superior threshold choice. In this chapter, the research questions are restated, and the corresponding answers briefly summarised.

RQ1 Can a non-trivial multivariate time series dataset be generated from physically motivated simulation models?

As shown in Chapter 4, it is indeed feasible to generate a synthetic dataset using simulation tools. Thanks to the controlled environment enabled by simulation, almost full control is gained over the generation procedure. As is typical, simulation allows for the possibility of faster-than-real-time computation, while also offering much more flexibility. This is especially evident in the case of the variable initial states like battery temperature and state of charge used in the powertrain anomaly time series benchmark (PATH) dataset. In the real world, this would require laborious charging/discharging and heating/cooling of the battery before any drive cycle, while in the case of a simulation only two scalar parameters need to be adjusted. For anomaly detection specifically, using a simulation model is associated with a unique set of benefits too. Recall Chapter 3, where the correctness of ground-truth label vectors for some publicly available datasets are questioned. A simulation model, in contrast, allows for precise timing for the start of anomalies and therefore ground-truth labels. It should

be noted, however, that even in simulation it can be difficult to pinpoint the end of an anomaly in the resulting data, as the anomalous behaviour may still be present but not observable due to slow dynamics of some internal processes, for example. With a simulation model, anomalies can also be generated with less bias and more realistically than manually tampering with the data, by provoking anomalous behaviour using, for instance, the change of a parameter before simulation. Lastly, a simulation model can, with some domain expertise, be extended to increase complexity and therefore generate data that is even closer to the real world. While the PATH dataset is based on a simulation model from the automotive domain, other simulation models from other domains could also be used to create a portfolio of different datasets, which would enrich the benchmarking aspect of the anomaly detection research field.

RQ2 Are satisfactory results obtainable when detecting anomalous behaviour in multivariate time series using completely unsupervised methods?

On the one hand, the answer to this research question may seem straightforward going by the results for the unsupervised version of the PATH dataset provided in Chapter 5.3. Recall that this version features a contaminated training subset with around 8 % unlabelled anomalous sequences and hence not only poses a challenge in terms of robust modelling of nominal behaviour but also in terms of threshold setting. Especially the latter problem manifests itself clearly, as truly unsupervised approaches, i.e. trained on unlabelled data and using the unsupervised threshold are far outmatched by their counterparts trained on unlabelled data but calibrated using the best possible threshold. It becomes clear that the unsupervised threshold is the weakest link in the anomaly detection algorithm, as it is highly sensitive to unlabelled anomalous data. But what if the training subset is not contaminated? These results further underline the high sensitivity of the threshold, as without contamination the results obtained using the unsupervised threshold exceed the previous ones by a large margin and feature a much smaller gap to their counterpart obtained using the best possible threshold. The results using the real-world dataset, which is also free of anomalies in the training subset, further contribute to this observation. Approaches fitted on a training subset with nominal only data, however, are classified as semi-supervised, and hence satisfactory results are not obtainable using truly unsupervised methods, highlighting the need for subsequent research on threshold choice in such cases.

RQ3 How well can a time series anomaly detection approach generalise to unseen data stemming from the same dynamic system?

To answer this research question, the PATH dataset is split into different cases where each is missing sequences with either low, medium or high average vehicle speeds or dynamism levels. A model that learns the underlying nominal behaviour of the dynamic system should in theory be able to generalise and represent unseen behaviour, as long as it is not in an extreme operational range. This can be a challenge, as the approach must be able to discern unseen but nominal behaviour from anomalous behaviour. The results show that, the absence of highly dynamic time series data and high average vehicle speed takes an especially large toll on the anomaly detection results, while generally, the opposite is true for low average speed and low dynamism data.

RQ4 Can active learning be harnessed to find a suitable threshold for completely unsupervised methods?

Recall the answer to RQ2. Considering the high sensitivity of the unsupervised threshold when faced with contaminated and unlabelled datasets, there is a clear need for a more robust threshold estimation method. Active learning can provide alleviation to this problem by intelligently querying a domain expert and estimating a threshold based on the labelled data. Clearly, such a procedure is no longer truly unsupervised, though that does not mean that it cannot find application in the real world. Furthermore, countless publications in the literature simply assume labelled data to be available when setting hyperparameters and thresholds, as is discussed in Chapter 3. In contrast, using active learning for setting a threshold forgoes this assumption and experiments associated attempt to simulate this challenge. Results show that, obtaining a threshold using active learning outperforms the unsupervised threshold by a large margin, depending on the query strategy employed. Compared to more traditional query strategies, the novel dissimilarity-based querying strategy outperforms in almost all cases, with particularly large margins in very low querying budget scenarios.

RQ5 How robust are active learning-based thresholds to mislabelling by the oracle?

The term of a domain expert is fairly loosely defined. Generally, it is interpreted as someone who is well versed in their application, though this can be at different levels depending on the individual. Furthermore, humans make mistakes and hence the impact of mislabelling on anomaly detection performance is investigated. This is especially interesting for practitioners to gauge how robust the estimation of thresholds using active learning can be in the real world. The experiments undertaken demonstrate that most of the results for all query strategies remain above the unsupervised

threshold, representing an improvement, even at fairly high mislabelling rates of three in ten, for instance. While promising, the results could still be improved to further approach those obtained using the best possible threshold.

Beyond the future work resulting from the research questions, further research should be dedicated to other aspects. These relate to benchmarking only, as it sets the foundation for measurable advancements in the future. While a set of metrics apt for online evaluation of discrete-sequence problems is proposed in this work, they still require further development, as outlined in Chapter 5. An adaptation of the improved range-based metrics [33] to discrete-sequence problems in addition to continuous-sequence problems could serve as a solution. In addition to that, approach complexity needs to be quantified as normalised for when presenting results. In the field of optimisation, this is often done by denoting performance in relation to function calls, for example. While the complexity of parametric approaches can be loosely described by their parameter count, two different architectures with the same number of parameters may have two entirely different runtimes on the same hardware or benefit from acceleration from certain hardware. Furthermore, the parameter count cannot be obtained for non-parametric approaches, which makes the comparison between a non-zero parameter approach with a non-parametric approach difficult. Once a strong foundation is set for benchmarking and associated procedures are widely agreed upon, advances benefiting the fields of medical field [1] and server machines [2] to water distribution [3] and treatment [4] can be achieved.

Bibliography

[1] George Moody and Roger Mark. "The Impact of the MIT-BIH Arrhythmia Database". In: *IEEE Engineering in Med and Biology* 20.3 (2001), pp. 45–50. DOI: 10.13026/C2F305.

[2] Ya Su et al. "Robust Anomaly Detection for Multivariate Time Series through Stochastic Recurrent Neural Network". In: *International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 2828–2837. DOI: 10.1145/3292500.3330672.

[3] Chuadhry Mujeeb Ahmed, Venkata Reddy Palleti, and Aditya P. Mathur. "WADI: A water distribution testbed for research in the design of secure cyber physical systems". In: *International Workshop on Cyber-physical Systems for Smart Water Networks*. 2017, pp. 25–28. DOI: 10.1145/3055366.3055375.

[4] Aditya P. Mathur and Nils Ole Tippenhauer. "SWaT: a water treatment testbed for research and training on ICS security". In: *International Workshop on Cyber-physical Systems for Smart Water Networks*. IEEE, 2016, pp. 31–36. DOI: 10.1109/CySWater.2016.7469060.

[5] Lucas Correia et al. "Online model-based anomaly detection in multivariate time series: Taxonomy, survey, research challenges and future directions". In: *Engineering Applications of Artificial Intelligence* 138 (2024), p. 109323. DOI: 10.1016/j.engappai.2024.109323.

[6] Lucas Correia et al. "PATH: A Discrete-sequence Dataset for Evaluating Online Unsupervised Anomaly Detection Approaches for Multivariate Time Series". In: *Big Data Research* (2025), p. 100573. DOI: 10.1016/j.bdr.2025.100573.

[7] Lucas Correia et al. "MA-VAE: Multi-Head Attention-Based Variational Autoencoder Approach for Anomaly Detection in Multivariate Time-Series Applied to Automotive Endurance Powertrain Testing". In: *Conference on Neural Computation Theory and Applications*. SciTePress, 2023, pp. 407–418. DOI: 10.5220/0012163100003595.

[8] Lucas Correia et al. "TeVAE: A Variational Autoencoder Approach for Discrete Online Anomaly Detection in Variable-State Multivariate Time-Series Data". In: *Computational Intelligence*. Vol. 1196. Series Title: Studies in Computational Intelligence. Springer Nature Switzerland, 2025, pp. 106–132. DOI: 10.1007/978-3-031-85252-7_7.

[9] Lucas Correia et al. "DQS: A Low-Budget Query Strategy for Enhancing Unsupervised Data-driven Anomaly Detection Approaches". In: *Journal of Big Data* (2026). DOI: 10.1186/s40537-026-01524-3.

[10] Bernd G Wagner et al. *REMAINING USEFUL LIFE IN COMPLEX MULTI-COMPONENT SYSTEMS: TAXONOMY, REVIEW, AND RESEARCH DIRECTIONS*. 2025. DOI: 10.36227/techrxiv.176288069.90065228/v1.

[11] George E. P. Box. *Time series analysis: forecasting and control*. 4th ed. Wiley series in probability and statistics. J. Wiley & Sons, 2008. ISBN: 978-0-470-27284-8.

[12] Peter J. Brockwell and Richard A. Davis. *Introduction to Time Series and Forecasting*. Springer Texts in Statistics. Springer International Publishing, 2016. DOI: 10.1007/978-3-319-29854-2.

[13] D. M. Hawkins. *Identification of Outliers*. Springer Netherlands, 1980. DOI: 10.1007/978-94-015-3994-4.

[14] Ane Blázquez-García et al. "A Review on Outlier/Anomaly Detection in Time Series Data". In: *ACM Computing Surveys* 54.3 (2021), pp. 1–33. DOI: 10.1145/3444690.

[15] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly Detection for Discrete Sequences: A Survey". In: *IEEE Transactions on Knowledge and Data Engineering* 24.5 (2012), pp. 823–839. DOI: 10.1109/TKDE.2010.235.

[16] Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. "Anomaly detection in time series: a comprehensive evaluation". In: *Proceedings of the VLDB Endowment* 15.9 (2022), pp. 1779–1797. DOI: 10.14778/3538598.3538602.

[17] S. Hochreiter and J. Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[18] Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. "Learning to Forget: Continual Prediction with LSTM". In: *Neural Computation* 12.10 (2000), pp. 2451–2471. DOI: 10.1162/089976600300015015.

[19] Kyunghyun Cho et al. "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: *Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2014, pp. 1724–1734. DOI: 10.3115/v1/d14-1179.

[20] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *International Conference on Learning Representations*. 2014.

[21] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. "Stochastic Backpropagation and Approximate Inference in Deep Generative Models". In: *International Conference on Machine Learning*. 2014, II–1278–II–1286. DOI: 10.5555/3044805.3045035.

[22] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. ISBN: 978-0-262-03561-3.

[23] Ashish Vaswani et al. “Attention Is All You Need”. In: *Conference on Neural Information Processing Systems*. 2017. DOI: 10.5555/3295222.3295349.

[24] John S. Bridle. “Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition”. In: *Neurocomputing* (1990), pp. 227–236.

[25] Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. “Neural Machine Translation by Jointly Learning to Align and Translate”. In: *International Conference on Learning Representations*. 2015.

[26] Francois Chollet. *Deep Learning with Python*. Manning Publications, 2021. ISBN: 978-1-61729-686-4.

[27] Xiaohui Zhou et al. “Boundary-Driven Active Learning for Anomaly Detection in Time Series Data Streams”. In: *International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2024, pp. 6135–6139. DOI: 10.1109/ICASSP48485.2024.10446524.

[28] T. K. Vintsyuk. “Speech discrimination by dynamic programming”. In: *Cybernetics* 4.1 (1972), pp. 52–57. DOI: 10.1007/BF01074755.

[29] H. Sakoe and S. Chiba. “Dynamic programming algorithm optimization for spoken word recognition”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26.1 (1978), pp. 43–49. DOI: 10.1109/TASSP.1978.1163055.

[30] Wannes Meert et al. *DTAIDistance*. 2020. DOI: 10.5281/ZENODO.7158824.

[31] Renjie Wu and Eamonn Keogh. “Current Time Series Anomaly Detection Benchmarks are Flawed and are Creating the Illusion of Progress”. In: *IEEE Transactions on Knowledge and Data Engineering* (2021), pp. 2421–2429. DOI: 10.1109/TKDE.2021.3112126.

[32] Kyle Hundman et al. “Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding”. In: *International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 387–395. DOI: 10.1145/3219819.3219845.

[33] Dennis Wagner et al. “TimeSeAD: Benchmarking Deep Multivariate Time-Series Anomaly Detection”. In: *Transactions on Machine Learning Research* (2023).

[34] Phillip Wenig, Sebastian Schmidl, and Thorsten Papenbrock. “Anomaly Detectors for Multivariate Time Series: The Proof of the Pudding is in the Eating”. In: *International Conference on Data Engineering*. IEEE, 2024, pp. 96–101. DOI: 10.1109/ICDEW61823.2024.00018.

[35] Phillip Wenig, Sebastian Schmidl, and Thorsten Papenbrock. “TimeEval: a benchmarking toolkit for time series anomaly detection algorithms”. In: *Proceedings of the VLDB Endowment* 15.12 (2022), pp. 3678–3681. DOI: 10.14778/3554821.3554873.

[36] David Baumgartner et al. "mTADS: Multivariate Time Series Anomaly Detection Benchmark Suites". In: *International Conference on Big Data*. IEEE, 2023, pp. 588–597. DOI: 10.1109/BigData59044.2023.10386980.

[37] Haowen Xu et al. "Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications". In: *Conference on World Wide Web*. ACM Press, 2018, pp. 187–196. DOI: 10.1145/3178876.3185996.

[38] Keval Doshi, Shatha Abudalou, and Yasin Yilmaz. "Reward Once, Penalize Once: Rectifying Time Series Anomaly Detection". In: *International Joint Conference on Neural Networks*. 2022, pp. 1–8. DOI: 10.1109/IJCNN55064.2022.9891913.

[39] Nesime Tatbul et al. "Precision and Recall for Time Series". In: *Conference on Neural Information Processing Systems*. 2018.

[40] Won-Seok Hwang et al. "Time-Series Aware Precision and Recall for Anomaly Detection: Considering Variety of Detection Result and Addressing Ambiguous Labeling". In: *International Conference on Information and Knowledge Management*. 2019, pp. 2241–2244. DOI: 10.1145/3357384.3358118.

[41] Astha Garg et al. "An Evaluation of Anomaly Detection and Diagnosis in Multivariate Time Series". In: *IEEE Transactions on Neural Networks and Learning Systems* 33.6 (2022), pp. 2508–2517. DOI: 10.1109/TNNLS.2021.3105827.

[42] Alexis Huet, Jose Manuel Navarro, and Dario Rossi. "Local Evaluation of Time Series Anomaly Detection Algorithms". In: *SIGKDD Conference on Knowledge Discovery and Data Mining*. 2022, pp. 635–645. DOI: 10.1145/3534678.3539339.

[43] Julien Audibert et al. "Do deep neural networks contribute to multivariate time series anomaly detection?" In: *Pattern Recognition* 132 (2022), p. 108945. DOI: 10.1016/j.patcog.2022.108945.

[44] John Paparrizos et al. "Volume under the surface: a new accuracy evaluation measure for time-series anomaly detection". In: *Proceedings of the VLDB Endowment* 15.11 (2022), pp. 2774–2787. DOI: 10.14778/3551793.3551830.

[45] Qinghua Liu and John Paparrizos. "The Elephant in the Room: Towards A Reliable Time-Series Anomaly Detection Benchmark". In: *Conference on Neural Information Processing Systems*. Vol. 37. 2024, pp. 108231–108261.

[46] Steven Yen, Melody Moh, and Teng-Sheng Moh. "CausalConvLSTM: Semi-Supervised Log Anomaly Detection Through Sequence Modeling". In: *International Conference On Machine Learning And Applications*. IEEE, 2019, pp. 1334–1341. DOI: 10.1109/ICMLA.2019.00217.

[47] Zhiqiang Que et al. "Real-Time Anomaly Detection for Flight Testing Using AutoEncoder and LSTM". In: *International Conference on Field-Programmable Technology*. IEEE, 2019, pp. 379–382. DOI: 10.1109/ICFPT47387.2019.00072.

[48] Mohsin Munir et al. "DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series". In: *IEEE Access* 7 (2019), pp. 1991–2005. DOI: 10.1109/ACCESS.2018.2886457.

[49] Pankaj Malhotra et al. "Long Short Term Memory Networks for Anomaly Detection in Time Series". In: *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. 2015, pp. 89–94. ISBN: 978-2-87419-049-0.

[50] Sucheta Chauhan and Lovekesh Vig. "Anomaly detection in ECG time signals via deep long short-term memory networks". In: *Data Science and Advanced Analytics*. IEEE, 2015, pp. 1–7. DOI: 10.1109/DSAA.2015.7344872.

[51] Pavel Filonov, Andrey Lavrentyev, and Artem Vorontsov. "Multivariate Industrial Time Series with Cyber-Attack Simulation: Fault Detection Using an LSTM-based Predictive Data Model". In: *Conference on Neural Information Processing Systems*. 2016.

[52] Pavel Filonov, Fedor Kitashov, and Andrey Lavrentyev. "RNN-based Early Cyber-Attack Detection for the Tennessee Eastman Process". In: *International Conference on Machine Learning*. 2017.

[53] Markus Thill et al. "Anomaly Detection in Electrocardiogram Readings with Stacked LSTM Networks". In: *Information Technologies - Applications and Theory*. 2019, pp. 17–25.

[54] Ahmad Idris Tambuwal and Daniel Neagu. "Deep Quantile Regression for Unsupervised Anomaly Detection in Time-Series". In: *SN Computer Science* 2.6 (2021), p. 475. DOI: 10.1007/s42979-021-00866-4.

[55] Jinwei Pan et al. "DUMA: Dual Mask for Multivariate Time Series Anomaly Detection". In: *IEEE Sensors Journal* 23.3 (2023), pp. 2433–2442. DOI: 10.1109/JSEN.2022.3225338.

[56] Feng Xia et al. "Coupled Attention Networks for Multivariate Time Series Anomaly Detection". In: *IEEE Transactions on Emerging Topics in Computing* 12.1 (2024), pp. 240–253. DOI: 10.1109/TETC.2023.3280577.

[57] Narendhar Gugulothu et al. "Sparse Neural Networks for Anomaly Detection in High-Dimensional Time Series". In: *International Joint Conference on Artificial Intelligence*. 2018.

[58] Ruei-Jie Hsieh, Jerry Chou, and Chih-Hsiang Ho. "Unsupervised Online Anomaly Detection on Multivariate Sensing Time Series Data for Smart Manufacturing". In: *Service-Oriented Computing and Applications*. 2019, pp. 90–97. DOI: 10.1109/SOCA.2019.00021.

[59] Julien Audibert et al. "USAD: UnSupervised Anomaly Detection on Multivariate Time Series". In: *SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, 2020, pp. 3395–3404. DOI: 10.1145/3394486.3403392.

[60] Barıs Bayram, Taha Berkay Duman, and Gökhan Ince. "Real time detection of acoustic anomalies in industrial processes using sequential autoencoders". In: *Expert Systems* 38.1 (2021), e12564. DOI: 10.1111/exsy.12564.

[61] Pankaj Malhotra et al. “LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection”. In: *International Conference on Machine Learning*. 2016.

[62] Hajar Homayouni et al. “An Autocorrelation-based LSTM-Autoencoder for Anomaly Detection on Time-Series Data”. In: *International Conference on Big Data*. IEEE, 2020, pp. 5068–5077. DOI: 10.1109/BigData50022.2020.9378192.

[63] Benjamin Lindemann, Nasser Jazdi, and Michael Weyrich. “Anomaly detection and prediction in discrete manufacturing based on cooperative LSTM networks”. In: *Conference on Automation Science and Engineering*. 2020, pp. 1003–1010. DOI: 10.1109/CASE48305.2020.9216855.

[64] H.D. Nguyen et al. “Forecasting and Anomaly Detection approaches using LSTM and LSTM Autoencoder techniques with the applications in supply chain management”. In: *International Journal of Information Management* 57 (2021), p. 102282. DOI: 10.1016/j.ijinfomgt.2020.102282.

[65] Susumu Naito et al. “Anomaly Detection for Multivariate Time Series on Large-scale Fluid Handling Plant Using Two-stage Autoencoder”. In: *International Conference on Data Mining Workshops*. 2021, pp. 542–551. DOI: 10.1109/ICDMW53433.2021.00072.

[66] Tung Kieu, Bin Yang, and Christian S. Jensen. “Outlier Detection for Multi-dimensional Time Series Using Deep Neural Networks”. In: *International Conference on Mobile Data Management*. IEEE, 2018, pp. 125–134. DOI: 10.1109/MDM.2018.00029.

[67] Tung Kieu et al. “Outlier Detection for Time Series with Recurrent Autoencoder Ensembles”. In: *International Joint Conference on Artificial Intelligence*. 2019, pp. 2725–2732. DOI: 10.24963/ijcai.2019/378.

[68] Tareq Tayeh et al. “An Attention-Based ConvLSTM Autoencoder with Dynamic Thresholding for Unsupervised Anomaly Detection in Multivariate Time Series”. In: *Machine Learning and Knowledge Extraction* 4.2 (2022), pp. 350–370. DOI: 10.3390/make4020015.

[69] Chuxu Zhang et al. “A Deep Neural Network for Unsupervised Anomaly Detection and Diagnosis in Multivariate Time Series Data”. In: *AAAI Conference on Artificial Intelligence*. 2019, pp. 1409–1416. DOI: 10.1609/aaai.v33i01.33011409.

[70] Gavneet Singh Chadha et al. “Deep Convolutional Clustering-Based Time Series Anomaly Detection”. In: *Sensors* 21.16 (2021), p. 5488. DOI: 10.3390/s21165488.

[71] Markus Thill et al. “Temporal convolutional autoencoder for unsupervised anomaly detection in time series”. In: *Applied Soft Computing* 112 (2021), p. 107751. DOI: 10.1016/j.asoc.2021.107751.

[72] Sawenbo Gong et al. "A Prediction-Augmented AutoEncoder for Multivariate Time Series Anomaly Detection". In: *Neural Information Processing*. Springer International Publishing, 2021, pp. 681–692. DOI: 10.1007/978-3-030-92185-9_56.

[73] Chunkai Zhang et al. "Reconstruct Anomaly to Normal: Adversarially Learned and Latent Vector-Constrained Autoencoder for Time-Series Anomaly Detection". In: *Pacific Rim International Conference on Artifical Inteligence*. Vol. 13032. 2021, pp. 515–529. DOI: 10.1007/978-3-030-89363-7_39.

[74] Daehyung Park, Yuuna Hoshi, and Charles C. Kemp. "A Multimodal Anomaly Detector for Robot-Assisted Feeding Using an LSTM-Based Variational Autoencoder". In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 1544–1551. DOI: 10.1109/LRA.2018.2801475.

[75] Kai Zhang et al. "Federated Variational Learning for Anomaly Detection in Multivariate Time Series". In: *International Performance, Computing, and Communications Conference*. 2021, pp. 1–9. DOI: 10.1109/IPCCC51483.2021.9679367.

[76] Dan Li et al. "MAD-GAN: Multivariate Anomaly Detection for Time Series Data with Generative Adversarial Networks". In: *International Conference on Artificial Neural Networks*. 2019, pp. 703–716. ISBN: 978-3-030-30490-4.

[77] Jin Fan et al. "LUAD: A lightweight unsupervised anomaly detection scheme for multivariate time series data". In: *Neurocomputing* 557 (2023), p. 126644. DOI: 10.1016/j.neucom.2023.126644.

[78] Xiaoxia Zhang et al. "ACVAE: A novel self-adversarial variational auto-encoder combined with contrast learning for time series anomaly detection". In: *Neural Networks* 171 (2024), pp. 383–395. DOI: 10.1016/j.neunet.2023.12.023.

[79] Yifan Guo et al. "Multidimensional Time Series Anomaly Detection: A GRU-based Gaussian Mixture Variational Autoencoder Approach". In: *Asian Conference on Machine Learning*. Vol. 95. 2018, pp. 97–112.

[80] Julian von Schleinitz et al. "VASP: An autoencoder-based approach for multivariate anomaly detection and robust time series prediction with application in motorsport". In: *Engineering Applications of Artificial Intelligence* 104 (2021), p. 104354. DOI: 10.1016/j.engappai.2021.104354.

[81] Longyuan Li et al. "Anomaly Detection of Time Series With Smoothness-Inducing Sequential Variational Auto-Encoder". In: *Transactions on Neural Networks and Learning Systems* 32.3 (2021), pp. 1177–1191. DOI: 10.1109/TNNLS.2020.2980749.

[82] Taesung Choi et al. "Multivariate Time-series Anomaly Detection using SeqVAE-CNN Hybrid Model". In: *International Conference on Information Networking*. 2022, pp. 250–253. DOI: 10.1109/ICOIN53446.2022.9687205.

[83] Bin Zhou et al. "BeatGAN: Anomalous Rhythm Detection using Adversarially Generated Time Series". In: *International Joint Conferences on Artificial Intelligence Organization*. 2019, pp. 4433–4439. DOI: 10.24963/ijcai.2019/616.

[84] Yeji Choi et al. "GAN-Based Anomaly Detection and Localization of Multivariate Time Series Data for Power Plant". In: *International Conference on Big Data and Smart Computing*. 2020, pp. 71–74. DOI: 10.1109/BigComp48618.2020.00-97.

[85] Alexander Geiger et al. "TadGAN: Time Series Anomaly Detection Using Generative Adversarial Networks". In: *International Conference on Big Data*. 2020, pp. 33–43. DOI: 10.1109/BigData50022.2020.9378139.

[86] Guangxuan Zhu et al. "A Novel LSTM-GAN Algorithm for Time Series Anomaly Detection". In: *Prognostics and System Health Management Conference*. 2019, pp. 1–6. DOI: 10.1109/PHM-Qingdao46334.2019.8942842.

[87] Yong Sun et al. "Time Series Anomaly Detection Based on GAN". In: *Social Networks Analysis, Management and Security*. 2019, pp. 375–382. DOI: 10.1109/SNAMS.2019.8931714.

[88] Hongwei Zhang et al. "Unsupervised Anomaly Detection in Multivariate Time Series through Transformer-based Variational Autoencoder". In: *Chinese Control and Decision Conference*. 2021, pp. 281–286. DOI: 10.1109/CCDC52312.2021.9601669.

[89] Shreshth Tuli, Giuliano Casale, and Nicholas R. Jennings. "TranAD: deep transformer networks for anomaly detection in multivariate time series data". In: *Very Large Databases*. 2022, pp. 1201–1214. DOI: 10.14778/3514061.3514067.

[90] Ling-rui Yu, Qiu-hong Lu, and Yang Xue. "DTAAD: Dual Tcn-attention networks for anomaly detection in multivariate time series data". In: *Knowledge-Based Systems* 295 (2024), p. 111849. DOI: 10.1016/j.knosys.2024.111849.

[91] Jiehui Xu et al. "Anomaly Transformer: Time Series Anomaly Detection with Association Discrepancy". In: *International Conference on Learning Representations*. 2022.

[92] Zekai Chen et al. "Learning Graph Structures with Transformer for Multivariate Time Series Anomaly Detection in IoT". In: *IEEE Internet of Things Journal* 9.12 (2021), pp. 9179–9189. DOI: 10.1109/JIOT.2021.3100509.

[93] Xixuan Wang et al. "Variational transformer-based anomaly detection approach for multivariate time series". In: *Measurement: Journal of the International Measurement Confederation* 191 (2022), p. 110791. DOI: 10.1016/j.measurement.2022.110791.

[94] Daniel Fährmann et al. "Lightweight Long Short-Term Memory Variational Auto-Encoder for Multivariate Time Series Anomaly Detection in Industrial Control Systems". In: *Sensors* 22.8 (2022), p. 2886. DOI: 10.3390/s22082886.

[95] Takuya Sakuma and Hiroki Matsutani. "An Area-Efficient Recurrent Neural Network Core for Unsupervised Time-Series Anomaly Detection". In: *IEICE Transactions on Electronics* E104.C.6 (2021), pp. 247–256. DOI: 10.1587/transele.2020LHP0003.

[96] Mathworks. *Build Full Electric Vehicle Model*. 2024. URL: https://de.mathworks.com/help/autoblks/ug/explore-the-electric-vehicle-reference-application.html (visited on 04/15/2024).

[97] Mathworks. *EV Battery Cooling System*. 2024. URL: https://de.mathworks.com/help/hydro/ug/ev-battery-cooling.html (visited on 04/15/2024).

[98] C.E. Shannon. "Communication in the Presence of Noise". In: *Proceedings of the IRE* 37.1 (1949), pp. 10–21. DOI: 10.1109/JRPROC.1949.232969.

[99] Hareesh Bahuleyan et al. "Variational Attention for Sequence-to-Sequence Models". In: *International Conference on Computational Linguistics*. Association for Computational Linguistics, 2018, pp. 1672–1682.

[100] Joao Pereira and Margarida Silveira. "Unsupervised Anomaly Detection in Energy Time Series Data Using Variational Recurrent Autoencoders with Attention". In: *International Conference on Machine Learning and Applications*. 2018, pp. 1275–1282. DOI: 10.1109/ICMLA.2018.00207.

[101] Tingting Chen et al. "Unsupervised Anomaly Detection of Industrial Robots Using Sliding-Window Convolutional Variational Autoencoder". In: *IEEE Access* 8 (2020), pp. 47072–47081. DOI: 10.1109/ACCESS.2020.2977892.

[102] Don S. Lemons. *An Introduction to Stochastic Processes in Physics*. Johns Hopkins University Press, 2002. DOI: 10.56021/9780801868665.

[103] Danilo Jimenez Rezende and Shakir Mohamed. "Variational Inference with Normalizing Flows". In: *International Conference on Machine Learning*. 2015, pp. 1530–1538. DOI: 10.5555/3045118.3045281.

[104] Sadaf Tafazoli and Eamonn Keogh. "Matrix Profile XXVIII: Discovering Multi-Dimensional Time Series Anomalies with K of N Anomaly Detection". In: *SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, 2023, pp. 685–693. DOI: 10.1137/1.9781611977653.

[105] Xuning Tang, Yihua Shi Astle, and Craig Freeman. "Deep Anomaly Detection with Ensemble-Based Active Learning". In: *International Conference on Big Data*. IEEE, 2020, pp. 1663–1670. DOI: 10.1109/BigData50022.2020.9378315.

[106] Tao Huang, Pengfei Chen, and Ruipeng Li. "A Semi-Supervised VAE Based Active Anomaly Detection Framework in Multivariate Time Series for Online Systems". In: *ACM Web Conference*. ACM, 2022, pp. 1797–1806. DOI: 10.1145/3485447.3511984.

[107] Wenlu Wang et al. "Active-MTSAD: Multivariate Time Series Anomaly Detection With Active Learning". In: *International Conference on Dependable Systems and Networks*. IEEE, 2022, pp. 263–274. DOI: 10.1109/DSN53405.2022.00036.

[108] Zhihan Li et al. "Situation-Aware Multivariate Time Series Anomaly Detection Through Active Learning and Contrast VAE-Based Models in Large Distributed Systems". In: *IEEE Journal on Selected Areas in Communications* 40.9 (2022), pp. 2746–2765. DOI: 10.1109/JSAC.2022.3191341.

[109] Rongwei Yu, Yong Wang, and Wang Wang. "AMAD: Active learning-based multivariate time series anomaly detection for large-scale IT systems". In: *Computers & Security* 137 (2024), p. 103603. DOI: 10.1016/j.cose.2023.103603.

[110] Adam Lundström, Mattias O'Nils, and Faisal Z. Qureshi. "An Interactive Threshold-Setting Procedure for Improved Multivariate Anomaly Detection in Time Series". In: *IEEE Access* 11 (2023), pp. 93898–93907. DOI: 10.1109/ACCESS.2023.3310653.

[111] Burr Settles. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Springer International Publishing, 2012. DOI: 10.1007/978-3-031-01560-1.

[112] Yao Wang et al. "Practical and White-Box Anomaly Detection through Unsupervised and Active Learning". In: *International Conference on Computer Communications and Networks*. IEEE, 2020, pp. 1–9. DOI: 10.1109/ICCCN49398.2020.9209704.

[113] Hamza Bodor, Thai V. Hoang, and Zonghua Zhang. "Little Help Makes a Big Difference: Leveraging Active Learning to Improve Unsupervised Time Series Anomaly Detection". In: *International Conference on Service Oriented Computing*. Vol. 13236. Springer International Publishing, 2022, pp. 165–176. DOI: 10.1007/978-3-031-14135-5_13.

[114] Jin Ning et al. "Deep Active Autoencoders for Outlier Detection". In: *Neural Processing Letters* 54.2 (2022), pp. 1399–1411. DOI: 10.1007/s11063-021-10687-4.

[115] Stefania Russo et al. "Active learning for anomaly detection in environmental data". In: *Environmental Modelling & Software* 134 (2020), p. 104869. DOI: 10.1016/j.envsoft.2020.104869.

[116] Martin Hervé De Beaulieu et al. "Unsupervised Remaining Useful Life Prediction through Long Range Health Index Estimation Based on Encoders-Decoders". In: *IFAC-PapersOnLine* 55.6 (2022), pp. 718–723. DOI: 10.1016/j.ifacol.2022.07.212.

[117] Martin Hervé De Beaulieu et al. "Unsupervised Prognostics Based on Deep Virtual Health Index Prediction". In: *PHM Society European Conference* 7.1 (2022), pp. 193–199. DOI: 10.36001/phme.2022.v7i1.3359.

Repositories

All source code, as well as the powertrain anomaly time series benchmark (PATH) dataset, are published under the MIT license. The source code can be found in GitHub repository under `github.com/lcs-crr/Thesis`. The PATH dataset in its various forms can be found in the Zenodo repository under `zenodo.org/records/13255120`.

List of Abbreviations

BiLSTM	bidirectional long short-term memory
DQS	dissimilarity-based query strategy
DTW	dynamic time warping
ELBO	evidence lower bound
EM	electric motor
FEV	full electric vehicle
FS	feedback strategy
GAN	generative adversarial network
GRU	gated recurrent unit
GutenTAG	good time series anomaly generator
HVB	high-voltage battery
LSTM	long short-term memory
LW-VAE	lightweight LSTM-VAE
MA	multi-head attention
MP	matrix profile
MS	multi-head self-attention
MSE	mean-squared error
MSL	Mars Science Laboratory
OA	OmniAnomaly
PATH	powertrain anomaly time series benchmark
PdM	predictive maintenance
PM	preventative maintenance
QS	query strategy
RNN	recurrent reural retworks
RQS	random-based query strategy

RTF	run to failure
RWM	reverse-window method
SISVAE	smoothness-inducing sequential VAE
SMAP	soil moisture active passive
SMD	server machine dataset
SoC	state of charge
SWaT	secure water treatment
TCN-AE	temporal convolutional autoencoder
TeVAE	temporal VAE
TQS	top-based query strategy
TSADIS	time series anomaly detection through incremental search
UQS	uncertainty-based query strategy
VAE	variational autoencoder
VASP	VAE-based selective prediction
VS	variational self-attention
VS-VAE	variational self-attention VAE
WADI	water distribution

Summary

This thesis investigates the feasibility of data-driven and truly unsupervised representation learning to time series anomaly detection problems. Such problems are relevant to the real world, as problems from it often lack available labels, for example due to prohibitively high costs. Therefore, advances in this field would benefit a variety of applications, though unfortunately such progress is currently hindered by the lack of reliable evaluation metrics and high-quality public datasets. This is further complicated by the fact that comparatively little work has been dedicated to contributions in benchmarking. Without metrics and datasets, it is difficult to objectively locate how far the current research has come and how robust claims in the literature really are.

To address this, this thesis contributes to the research field by providing a novel and non-trivial time series dataset, which is generated with state-of-the-art simulation tools. While technically a synthetic dataset, the simulation model it is generated with is real-world inspired, meaning that much of the complexity of a real-world system is modelled. Harnessing simulation to generate data also allows for a greater level of granularity over the anomaly generation process than would otherwise be possible during collection of a real-world dataset.

Another aspect hindering scientific advances from being applied to real-world problems is also the rather loose definition of unsupervised methods in literature. Many publications revolving around time series anomaly detection may claim to be unsupervised, though further inspection reveals that at least one component within the proposed methodology, usually the threshold used to isolate anomalous data points, assumes the presence of labelled data. Therefore, this thesis also investigates how well *truly* unsupervised methods perform on real-world and real-world-like datasets, not only in terms of anomaly detection performance, but also in terms of generalisation capability. Results show that, when training subset is contaminated with anomalies,

truly unsupervised methods, i.e. using the unsupervised threshold, struggle, with the best approach achieving $F_1 = 0.06$. This can be largely attributed to the threshold choice, as setting it to a value that maximises the F_1 score, shows that some approaches achieve much more respectable results of up to $F_1 = 0.58$. When the training subset is cleansed of anomalous data, using the unsupervised threshold yields much better calibrated approaches compared to a contaminated training subset, achieving up to $F_1 = 0.67$. Here, the ideal threshold yields better results with an F_1 score of up to 0.86; a gap much smaller than previously with a contaminated training subset. Overall it becomes clear, that modelling alone does not dictate how well a model performs, since if it is paired with a poorly chosen threshold, the potential of the resulting approach is masked. Furthermore, the experiment investigating the generalisation capability of a model shows that, for the automotive powertrain data used, the model is able to generalise better over the data if the training subset features especially dynamic or high vehicle-speed time series.

Instead of assuming the availability of labelled data, this thesis investigates the feasibility of active learning to further enhance truly unsupervised anomaly detection methods. The proposed framework intelligently collects sequences most dissimilar to one another in an effort to maximise diversity. In real-world problems, these sequences can be presented to a domain expert who can label them, thereby forming a small labelled dataset which can be used for further threshold optimisation. The novel query strategy proposed shows that, especially in low query budget scenarios, it outperforms approaches calibrated with an unsupervised threshold almost across the board, almost matching the results obtained with the best possible threshold. Since even domain experts can make mistakes, this work also considers the effect of mislabelling on the anomaly detection performance, illustrating the robustness of active learning. Even in the most extreme case investigated, where the domain expert was simulated to mislabel almost a third of the queries, approaches calibrated with the resulting active learning-based threshold still outperformed their unsupervised counterpart, indicating a level of robustness to human error.

Moreover, this work also shows how the same data-driven modelling procedure can be applied to other time series problems like predictive maintenance. In industrial settings, predictive maintenance aims to inform about the right time for component changes due to deterioration and wear. This is in contrast to more traditional practices of preventive maintenance, which involves swapping components at an early stage, leading to unused remaining useful life within the component, and run to failure, which involves running a component until it is no longer usable, which can damage

other components in the process. For a case-study from a real-world automotive test bench, the data-driven model was able to detect significant deterioration of the testee 54 min earlier than the currently implemented rule-based monitoring system.

Lastly, this work concludes that several aspects of the research field deserve future work to be dedicated to them. Benchmarking especially requires further contributions, not only in terms of dataset diversity but also in terms of general and flexible evaluation metrics, in addition to the need for the quantification of approach complexity. Additionally, the experiments involving active learning show that robustness to mislabelling can be further improved, which is of large interest to industrial applications. For cases where a domain expert is not available, future work should also seek to provide improved ways of obtaining truly unsupervised thresholds.

Samenvatting

Dit proefschrift onderzoekt de haalbaarheid van datagedreven en werkelijk ongesuperviseerde representatie-leermethoden voor anomaliedetectie in tijdreeksen. Dergelijke vraagstukken zijn relevant voor de echte wereld, aangezien problemen daar vaak een gebrek aan beschikbare labels hebben, bijvoorbeeld door buitensporig hoge kosten. Daarom zouden vorderingen op dit gebied ten goede komen aan een verscheidenheid aan toepassingen, hoewel dergelijke vooruitgang momenteel helaas wordt gehinderd door het gebrek aan betrouwbare gegevensstatistieken en publieke datasets van hoge kwaliteit. Dit wordt verder bemoeilijkt door het feit dat er naar verhouding weinig onderzoek is gewijd aan bijdragen op het gebied van benchmarking. Zonder statistieken en datasets is het moeilijk om objectief vast te stellen hoe ver het huidige onderzoek gevorderd is en hoe robuust de claims in de literatuur werkelijk zijn.

Om dit aan te pakken, draagt deze scriptie bij aan het onderzoeksveld door een nieuwe en niet-triviale tijdreeks-dataset aan te bieden, die is gegenereerd met geavanceerde simulatietools. Hoewel het technisch gezien een synthetische dataset is, is het simulatiemodel waarmee deze is gegenereerd geïnspireerd op de echte wereld, wat betekent dat veel van de complexiteit van een systeem uit de praktijk is gemodelleerd. Het aanwenden van simulatie om data te genereren maakt bovendien een hogere mate van granulariteit over het proces van anomaliegeneratie mogelijk dan anders mogelijk zou zijn tijdens het verzamelen van een dataset uit de echte wereld.

Een ander aspect dat de toepassing van wetenschappelijke vooruitgang op praktijkproblemen belemmert, is de vrij losse definitie van ongesuperviseerde methoden in de literatuur. Veel publicaties over anomaliedetectie in tijdreeksen beweren ongesuperviseerd te zijn, terwijl bij nadere inspectie blijkt dat ten minste één component binnen de voorgestelde methodologie — meestal de drempelwaarde die wordt gebruikt om afwijkende datapunten te isoleren — uitgaat van de aanwezigheid van gelabelde data. Daarom onderzoekt deze scriptie ook hoe goed *volledig* ongesuperviseerde me-

thoden presteren op datasets uit de echte wereld (en daarop gelijkende datasets), niet alleen in termen van anomaliedetectie-prestaties, maar ook in termen van generalisatievermogen. Resultaten tonen aan dat, wanneer de trainingssubset gecontamineerd is met anomalieën, werkelijk ongesuperviseerde methoden, d.w.z. met gebruik van de ongesuperviseerde drempelwaarde, moeite hebben, waarbij de beste aanpak een $F_1 = 0.06$ behaalt. Dit kan grotendeels worden toegeschreven aan de keuze van de drempelwaarde, aangezien het instellen ervan op een waarde die de F_1-score maximaliseert, aantoont dat sommige aanpakken veel respectabelere resultaten behalen, tot wel $F_1 = 0.58$. Wanneer de trainingssubset ontdaan is van anomale data, levert het gebruik van de ongesuperviseerde drempelwaarde veel beter gekalibreerde aanpakken op in vergelijking met een gecontamineerde trainingssubset, waarbij waarden tot $F_1 = 0.67$ worden behaald. Hier levert de ideale drempelwaarde betere resultaten op, met een F_1-score tot wel 0.86; een verschil dat veel kleiner is dan eerder bij een gecontamineerde trainingssubset. Over het geheel genomen wordt duidelijk dat modellering alleen niet bepaalt hoe goed een model presteert, aangezien het potentieel van de resulterende aanpak wordt gemaskeerd wanneer deze gepaard gaat met een slecht gekozen drempelwaarde. Bovendien toont het experiment naar het generalisatievermogen van een model aan dat, voor de gebruikte automotive aandrijflijndata, het model beter over de data kan generaliseren wanneer de trainingssubset bijzonder dynamische tijdreeksen of tijdreeksen met hoge voertuigsnelheid bevat.

In plaats van uit te gaan van de beschikbaarheid van gelabelde data, onderzoekt deze scriptie de haalbaarheid van active learning om werkelijk ongesuperviseerde anomaliedetectie-methoden verder te verbeteren. Het voorgestelde raamwerk verzamelt op intelligente wijze sequenties die het meest van elkaar verschillen om de diversiteit verder te maximaliseren. Bij praktijkproblemen kunnen deze sequenties worden voorgelegd aan een domeinexpert die ze kan labelen, waardoor een kleine gelabelde dataset ontstaat die kan worden gebruikt voor verdere optimalisatie van de drempelwaarde. De voorgestelde nieuwe query-strategie toont aan dat zij, met name in scenario's met een laag querybudget, vrijwel over de gehele linie beter presteert dan aanpakken die gekalibreerd zijn met een ongesuperviseerde drempelwaarde, en daarbij de resultaten die met de best mogelijke drempelwaarde worden behaald vrijwel evenaart. Aangezien zelfs domeinexperts fouten kunnen maken, beschouwt dit werk ook het effect van foutieve labeling op de prestaties van de anomaliedetectie, waarmee de robuustheid van active learning wordt aangetoond. Zelfs in het meest extreme onderzochte geval, waarin gesimuleerd werd dat de domeinexpert bijna een derde van de queries verkeerd labelde, presteerden aanpakken die gekalibreerd waren met de

resulterende op active learning gebaseerde drempelwaarde nog steeds beter dan hun ongesuperviseerde tegenhanger, wat duidt op een zekere mate van robuustheid tegen menselijke fouten.

Bovendien laat dit werk zien hoe dezelfde datagedreven modelleringsprocedure kan worden toegepast op andere tijdreeksproblemen, zoals predictief onderhoud. In een industriële omgeving streeft predictief onderhoud ernaar te informeren over het juiste moment voor componentvervanging als gevolg van verslechtering en slijtage. Dit staat in contrast met de meer traditionele praktijken van preventief onderhoud, waarbij componenten in een vroeg stadium worden vervangen, wat leidt tot ongebruikte resterende levensduur, en run-to-failure, waarbij een component wordt gebruikt totdat deze niet langer bruikbaar is, wat in het proces andere componenten kan beschadigen. Zelfs in het meest extreme onderzochte geval, waarin gesimuleerd werd dat de domeinexpert bijna een derde van de queries verkeerd labelde, presteerden aanpakken die gekalibreerd waren met de resulterende op active learning gebaseerde drempelwaarde nog steeds beter dan hun ongesuperviseerde tegenhanger, wat duidt op een zekere mate van robuustheid tegen menselijke fouten.

Ten slotte concludeert dit werk dat verschillende aspecten van het onderzoeksveld toekomstig onderzoek verdienen. Vooral benchmarking vereist verdere bijdragen, niet alleen wat betreft de diversiteit van datasets, maar ook in termen van algemene en flexibele evaluatiestatistieken, naast de noodzaak voor het kwantificeren van de complexiteit van de benadering. Daarnaast laten de experimenten met active learning zien dat de robuustheid tegen foutieve labeling verder kan worden verbeterd, wat van groot belang is voor industriële toepassingen. Voor gevallen waarin geen domeinexpert beschikbaar is, zou toekomstig onderzoek ook moeten zoeken naar verbeterde manieren om werkelijk ongesuperviseerde drempelwaarden te verkrijgen.

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my academic supervisors, Prof. Dr. Thomas Bäck and Dr. Anna Kononova. Thank you for your steadfast oversight and your commitment to open science, which set the tone for how this research was conducted. Your attention to detail and your high standards sharpened both my thinking and my writing in ways I will carry forward.

I would also like to extend my deepest gratitude to Dr. Jan-Christoph Goos and Dr. Philipp Klein. Together with Prof. Bäck and Dr. Kononova, they played a central role in shaping the idea that eventually became this thesis, contributing invaluable perspective from real-world applications. Their day-to-day support was equally important, particularly in bridging the gap between current test bench challenges and my research, and in helping me understand how to best harness the data available to us. I would also like to mention my colleagues Jochen Stadler, Jonas Scheiffele, Daniel Scheuerle and Robin Müller, whose company and support made my daily work something I looked forward to every day.

I am grateful, too, for my network with other PhD students. Although few of the wonderful people I encountered there worked in a related research area, the network proved to be a reliable source of support during setbacks and a welcome outlet for frustration. The regular roundtables and excursions reminded me that there was a wider community navigating the same uncertainties, and that the PhD journey, for all its solitary moments, need not be faced alone. The network also gave me something I did not expect: several lasting friendships, including with Dr. Frauke Langer, Dr. Julian Schneider, Tim Brock and Jeroen Welles.

This thesis benefited greatly from the contributions of a number of talented and dedicated students. Vishwesh Bhagwat, Abdullah Kuloglu, Halime Cengiz and Lena Wesseln each brought fresh ideas and new directions to the research, and supervising them was a delight.

Working alongside Bernd Wagner and Qi Huang on a joint paper offered a different kind of learning, one centred on communication, collaboration and the art of finding common ground, which I am equally thankful for.

Finally, I wish to thank Ruben Correia, Paula Correia, Manuel Correia and Eva Krämer, who never lost faith in me and encouraged me to persevere, even in the moments when I was certain that continuing my PhD was not the right path. I owe a particular debt to my parents, Manuel and Paula, who laid the educational foundations that ultimately made it possible for me to pursue the career I have chosen.

About the Author

Lucas Ferreira Correia, born in 1998 in Braga, Portugal, completed his CIE A-levels at the Townshend International School in Czechia in 2016, after which he started his higher education path. In 2021 Lucas received his integrated master's degree in Aeronautical Engineering from the University of Glasgow, United Kingdom. That year, he started as an external PhD student at the Leiden Institute of Advanced Computer Science in the Netherlands, under the supervision of Dr. Anna Kononova and Prof. Dr. Thomas Bäck. During his PhD studies, he took a course in scientific conduct. The research was conducted as a member of the Efficient Heuristic Optimisation (EcHO) group within the Natural Computing cluster at LIACS. In 2025, he started working as a data scientist at the Liebherr-Digital Development Center in Ulm, Germany.