PATH: a discrete-sequence dataset for evaluating online unsupervised anomaly detection approaches for multivariate time series

Ferreira Correia, L.; Goos, J.C.; Bäck, T.H.W.; Kononova, A.V.

Big Data Research 42 (2025) 100573

Contents lists available at ScienceDirect

Big Data Research

journal homepage: www.elsevier.com/locate/bdr

PATH: A discrete-sequence dataset for evaluating online unsupervised anomaly detection approaches for multivariate time series

Lucas Correia [a,b,*], Jan-Christoph Goos [b], Thomas Bäck [a], Anna V. Kononova [a]

[a] LIACS, Leiden University, Einsteinweg 55, 2333 CC, Leiden, the Netherlands
[b] Mercedes-Benz AG, Mercedesstraße 120, 70372, Stuttgart, Germany

ARTICLE INFO

Keywords:
Anomaly detection
Dataset
Time series
Discrete-sequence
Multivariate
Unsupervised

ABSTRACT

Benchmarking anomaly detection approaches for multivariate time series is a challenging task due to a lack of high-quality datasets. Current publicly available datasets are too small, not diverse and feature trivial anomalies, which hinders measurable progress in this research area. We propose a solution: a diverse, extensive, and non-trivial dataset generated via state-of-the-art simulation tools that reflects realistic behaviour of an automotive powertrain, including its multivariate, dynamic and variable-state properties. Additionally, our dataset represents a discrete-sequence problem, which remains unaddressed by previously-proposed solutions in literature. To cater for both unsupervised and semi-supervised anomaly detection settings, as well as time series generation and forecasting, we make different versions of the dataset available, where training and test subsets are offered in contaminated and clean versions, depending on the task. We also provide baseline results from a selection of approaches based on deterministic and variational autoencoders, as well as a non-parametric approach. As expected, the baseline experimentation shows that the approaches trained on the semi-supervised version of the dataset outperform their unsupervised counterparts, highlighting a need for approaches more robust to contaminated training data. Furthermore, results show that the threshold used can have a large influence on detection performance, hence more work needs to be invested in methods to find a suitable threshold without the need for labelled data.

1. Introduction

As the digitisation of industrial processes progresses, more and more data is recorded. Ensuring this data is representative of the process is important, as 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 cardiology [1] and server metrics monitoring [2] to water systems [3–5] and traffic analysis [6–8]. Note that, we use *time series* and *sequences* synonymously throughout this paper.

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. An arbitrary time series S can be univariate, i.e. $S \in \mathbb{R}^T$, or multivariate, i.e. $S \in \mathbb{R}^{T \times d}$, 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 solely possess a temporal correlation, i.e. along the time axis, whereas multivariate time series can also contain correlation along the feature axis.

Detecting anomalous behaviour in time series is referred to *time series anomaly detection*, which can be split into two main areas: *continuous- and discrete-sequence* [9], where the former is the only type addressed in public datasets. Continuous-sequence problems are defined as detecting anomalies in a process that runs for a continuous time period without breaks. Typically, the test subset D^{test} in the dataset D in a continuous-sequence problem consists of a singular multivariate time series composed of multiple nominal and anomalous sub-sequences, i.e. $D^{\text{test}} = \{S_1\}$. In this work, we use *nominal* as a synonym for *normal* or *anomaly-free* to avoid confusion with Gaussian distributions. Discrete-sequence anomaly detection, in contrast, is defined as detecting anomalies in N chunks of processes that happen independently of each other. Discrete-sequence problems include, for example, automotive test benches, where several tests may occur sequentially but are

* Corresponding author.
E-mail addresses: l.ferreira.correia@liacs.leidenuniv.nl (L. Correia), a.kononova@liacs.leidenuniv.nl (A.V. Kononova).

https://doi.org/10.1016/j.bdr.2025.100573
Received 8 April 2025; Accepted 2 November 2025

Available online 7 November 2025

not temporally contiguous and hence provide a time series for each test, i.e. $D^{\text{test}} = \{S_1, ..., S_n, ..., S_N\}$ and $N > 1$. 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 or only partly. Depending on the system, the time series data may also feature *variable states*, meaning the recorded signals appear slightly different if certain external conditions change but are still considered nominal. One example of a variable-state system is a battery, where the voltage response to current changes depending on states like the battery temperature and the battery state of charge (SoC). A problem involving such a system requires the distinction between behaviour changes due to different states and behaviour changes due to an anomaly, further complicating detection.

In addition to that, detecting anomalous behaviour in a timely manner is also advantageous because the source of anomalous behaviour may bring about damage to said system. Problems where the detection delay plays a role require evaluation of the time series before it ends and are referred to as *online* time series anomaly detection.

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 the literature. This is most likely because, in real-world problems, possible anomaly types and how they manifest themselves in the data are rarely known a priori. Moreover, labelling data is expensive, which is why unsupervised and semi-supervised anomaly detection are more prevalent in literature and relevant to the real world. *Unsupervised* anomaly detection is independent of any labels, i.e. any available data for model training contains both anomalous and nominal time series, and it is not known which is which [10]. In contrast, *semi-supervised* anomaly detection can be considered a more relaxed setting, where anomalous time series are absent from the training subset [10]. In the real world, semi-supervised problems still require 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. [11], 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 we challenge in this paper.

Our contribution is 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 detection, we provide versions for semi-supervised anomaly detection and time series generation and forecasting as well. Despite the data being generated using simulation, the electric vehicle simulation model is strongly motivated by the real world and is therefore complex and variable-state.

This paper is structured as follows. First, we introduce the related work in the area of benchmarking time series anomaly detection approaches. It includes discussion on the datasets used to evaluate time series anomaly detection approaches in the past, and a summary of the work dedicated to outlining the status quo in benchmarking time series anomaly detection approaches. Then, we introduce the PATH dataset in detail, outlining the generation process and a few benchmarking considerations. Following that, we provide some baseline results for a selection of deep learning-based models, as well as a non-parametric approach. Finally, we conclude our work and outline an outlook on future work. The source code corresponding to this paper and the simulation model can be found on Github, and the dataset can be downloaded from Zenodo [12].

Table 1
Key properties of the most popular datasets, where d_D refers to the number of features of the time series signals in the dataset, DS to whether it is a discrete-sequence dataset, $\sum |S_n|$ to the number of test time steps and $\%A$ to the number of anomalous time steps in relation to all test time steps. GutenTAG varies in d_D depending on test time series. The number of time steps given for the test subset in PATH is the average across all folds.

Name	d_D	DS	$\sum_{}^{N} \lvert S_n \rvert$	$\%A$
SWaT	51	✗	449,919	12 %
WADI	127	✗	17,287	6 %
SMAP	25	✗	427,617	13 %
MSL	55	✗	73,729	11 %
SMD	38	✗	708,420	4 %
GutenTAG	2 – 20	✗	240,000	1 %
mTADS	4	✗	2,396,000	<1 %
Ours	16	✓	14,341,432	7 %

2. Related work

2.1. Publicly available datasets

Over the last few years, five benchmark datasets have emerged as by far the most popular, with at least one of them being cited in the vast majority of publications on multivariate time series anomaly detection. A summary of the properties of these datasets is shown in Table 1.

The MSL [13], SMAP [13], and SMD [2] have already been thoroughly analysed by Wu and Keogh [14], who point out several *issues* with the datasets. The first issue observed in the datasets is triviality, defined by being solvable using so-called *one-line code*, such as the moving standard deviation over a subset of the dataset features. Moreover, all of them suffer from what Wu and Keogh [14] have called unrealistic anomaly density, meaning that they have sub-sequences with a very high anomaly share and hence do not match the assumption that anomalies are rare events. In addition to that, Wu and Keogh [14] suspect possible mislabelling present in the MSL dataset. They base their suspicion on the fact that the dataset contains sub-sequences with static behaviour in an evidently dynamic channel, which is labelled as nominal. Additionally, while the SMAP and MSL datasets are technically multivariate, the channels are not synchronised with each other and hence each channel needs to be modelled on its own.

Many of the issues pointed out by Wu and Keogh [14] can also be extended to the SWaT and WADI datasets, as thoroughly discussed by Wagner et al. [15] and Correia et al. [9]. Both datasets have multiple features that are constant throughout the training and testing subsets, while WADI has some features with missing values. This leads to ambiguity when benchmarking, as some may choose to omit these features while others may not. Furthermore, 65% of the anomalous time steps in the SWaT dataset can be detected by simply inspecting one feature.

Additionally, Wenig et al. [16] 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.

Wu and Keogh [14] have caused a shift towards more transparency when benchmarking time series anomaly detection methods. Since then, some contributions have been made to address the aforementioned issues, also shown in Table 1.

As part of the TimeEval framework, Wenig et al. [17] 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 ECG-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 [11] 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.

mTADS [18] is a collection of two types of datasets, one completely synthetic 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's 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.

Certain real-world applications like automotive testing present *complexities* previously unseen in public datasets. As outlined by Correia et al. [19], such applications comprise much more diverse discrete-sequence datasets, owed to the presence of both highly dynamic and mostly static features, as well as variable states. The presence of variable states leads to features exhibiting a different pattern depending on the time it is observed. In the context of automotive testing, an example of such a feature would be the state of charge of a battery, which discharges with time and hence shows different behaviour for the same test done twice in a row.

Hence, we construct a new high-quality dataset that features non-trivial anomalies caused by pre-simulation model changes and that mostly reflects real-world complexity. Additionally, as a discrete-sequence dataset, it represents a contribution to an underrepresented problem type that often occurs in the real world. Since the dataset is generated using simulation, it also strikes the balance between real-world relevance and control over anomalous behaviour [18].

2.2. Doubts regarding applicability of deep learning

Recently, there has been growing doubt on whether deep learning (DL) algorithms are definitively the better choice for time series anomaly detection. For the purpose of this publication, classical methods refer to all approaches not based on DL, including non-parametric and statistical approaches, as well as simpler machine learning methods like clustering.

Wu and Keogh [14] claim that the superiority of DL in anomaly detection is assumed to be a given, despite a lack of clear evidence for the need for DL. They stress that existing classical methods should be considered, given their generally simpler and faster nature.

To investigate the comparative performance of classical methods and DL-based methods, Audibert et al. [20] analyse a variety of different models on five of the most popular benchmark datasets, shown in Table 1. They conclude that, across the datasets considered, there is no algorithm that dominates all the other ones, arguing that there is no reason to omit classical methods from benchmarking.

Rewicki et al. [21] also conduct a comparative study of classical and DL-based methods, though on the UCR Anomaly Archive benchmark proposed by Wu and Keogh [14], which exclusively contains univariate time series and therefore lacks correlations between channels present in multivariate time series. They conclude that classical methods perform better than their DL counterparts, although this is to be expected given the simpler, univariate nature of the dataset.

Table 2
Signals included in the PATH dataset, along with their physical units and persistent indices.

Index	Signal Name	Unit
1	Motor Speed	$\mathrm{rad\,s^{-1}}$
2	Motor Torque	N m
3	Axle Torque Front	N m
4	Axle Torque Rear	N m
5	Battery SoC	%
6	Battery Current	A
7	Battery Power	W
8	Axle Force Front	N
9	Axle Force Rear	N
10	Axle Speed Front	$\mathrm{rad\,s^{-1}}$
11	Axle Speed Rear	$\mathrm{rad\,s^{-1}}$
12	Accelerator Pedal	-
13	Brake Pedal	-
14	Battery Temperature	°C
15	Cooling Pump Power	W
16	Refrigerator Power	W

While the findings and doubts of the above-mentioned are valid, they are limited by the lack of large, high-quality multivariate datasets. In this paper, we purposefully include results from a state-of-the-art classical method to find out whether doubts on DL are still justified for extensive and complex real-world-inspired datasets. See Section 4 for results and discussion.

3. Proposed dataset

3.1. Simulation model

To create an extensive and diverse dataset, we propose to use a physically inspired model, from which we can generate data using simulation. MathWorks offers reference models for a variety of dynamic systems, one of which is the full electric vehicle (FEV) model[1] from the powertrain blockset in Simulink. This choice is based on our familiarity with the domain, as generating data blindly without any background may lead to systematic errors. The FEV model offered by MathWorks 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 Fig. 1.

To represent system behaviour, $d_D = 16$ signals are chosen to be logged during simulation and are summarised in Table 2. We chose these signals based on domain 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, a subset is eliminated due to their unrealistic nature, e.g. high linearity or duplicity, as many cycles are present as sub-sequences in others. Our analysis shows that, for example, the presence of the FTP72 drive cycle within FTP75 or the presence of the LA92Short drive cycle 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 3, 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.

[1] https://de.mathworks.com/help/autoblks/ug/explore-the-electric-vehicle-reference-application.html, last accessed: 24.03.2025.

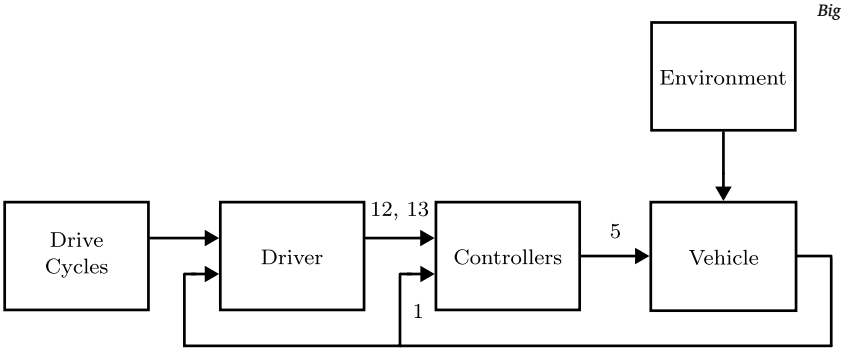

Fig. 1. Simplified schematic of the FEV model used for the generation of the PATH dataset. Numbers represent the indices of signal flow, reference is shown in Table 2.

Table 3
Drive cycles used for the PATH dataset 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

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 2, 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, electric motor speed and battery signals to calculate request signals for the powertrain, like the required electric motor torque and the brake signal, as well as battery management system signals like the battery SoC, index 5 in Table 2. Electric vehicles are capable of regenerative braking, meaning, the electric motor is used to decelerate the vehicle by acting as a generator, thereby charging the battery 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 battery SoC, and the environment block, as well as from each other, as shown in Fig. 2. The electric plant subsystem outputs the electric motor torque, the battery current and power, the battery temperature and the cooling pump and refrigerator powers, which correspond to indices 2, 6, 7, 14, 15 and 16 in Table 2, respectively. The electric motor torque is input into the drivetrain subsystem, which in turn outputs the electric motor speed, and front and rear axle torques, forces and speeds, corresponding to indices 1, 3, 4, 8, 9, 10, 11 in Table 2, respectively. The motor 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 paper. Readers interested in more detail can refer to the Simulink model available in the provided repository. By default, the battery model features a static temperature model, however, to increase system complexity, a dynamic temperature model is added to the FEV model. The model used is the EV Battery Cooling System,[2] 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 grade and coefficient of friction can be set 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 this, 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 simulations never take longer than 20 minutes, and hence one hour is considered sufficient for problem-free simulations.

3.2. Dataset generation

To generate a dataset that is not only extensive but also diverse, 100 simulations have been undertaken for each of the 33 drive cycles, each with random initial battery temperatures and battery SoCs. At this stage, all model properties have been left as default, and hence all simulation results have been considered *nominal*. For each simulation, the two states (battery temperatures and battery SoCs) have been sampled from uniform distributions $\mathcal{U}(10\,°\mathrm{C}, 30\,°\mathrm{C})$ and $\mathcal{U}(10\,\%, 100\,\%)$, respectively, to ensure no bias is introduced. Sampling from uniform distributions also reduces the effectiveness of simple threshold and control chart methods because the battery temperature and state of charge, but also, by extension, 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 % have been removed, as very

[2] https://de.mathworks.com/help/hydro/ug/ev-battery-cooling.html, last accessed: 24.03.2025.

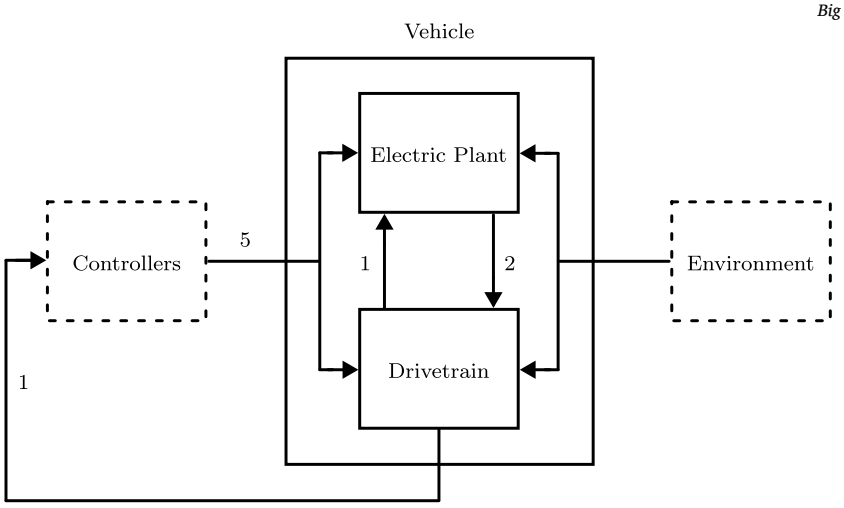

Fig. 2. A more detailed schematic of the vehicle model depicted in Fig. 1. Numbers represent the indices of signal flow, reference is shown in Table 2. Output signals of the vehicle model, which are not fed back into other subsystems, are not shown, for simplicity.

Table 4
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 Motor Torque Request	32	33
Increased Wheel Diameter	0	33
Increased Driver Reaction Time	0	33

low values have been observed to result in abnormal behaviour. After simulation, $N_n = 3273$ highly diverse and unique nominal time series have been collected. For illustration purposes, a nominal time series is plotted in Fig. 3.

For the generation of *anomalous* time series, six types of anomalies have been considered. Some types can occur as both sub-sequence anomalies and sequence anomalies [9], while others only in sequence anomaly form, due to simulation model limitations. To better distinguish the two anomaly forms, we refer to sequence anomalies as *whole-sequence* anomalies henceforth. The distribution of sub-sequence and whole-sequence anomalies across the different anomaly types is shown in Table 4. Anomalies are caused by changing 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 dataset [22], which eliminates any bias. We ran all simulations with a fixed seed of 1.

For the first kind of anomaly, we turn the regenerative braking off, which leads to visibly different motor and axle torques, as well as battery current and power, as these can no longer assume negative values. When regenerative braking is off, the battery SoC 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 braking motor torque. 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 drive cycle in question, see Table 3. This statistical distribution is used for all sub-sequence anomaly types. One of the anomalous time series for the CADC130 drive cycle and its control counterpart are plotted in Fig. 4.

In the case of the next anomaly type, we introduce a headwind of $5\,\mathrm{m\,s^{-1}}$ to the model. This headwind acts as a force on the frontal area of the vehicle and needs to be overcome to maintain the target vehicle speed by using the accelerator pedal more than the norm, which leads to higher motor and axle torques and therefore axle forces. The higher motor torque requires a higher battery 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 counterpart are plotted in Fig. 5.

Following that, we reduce the displacement of the cooling pump by 10 % to simulate another anomaly type. Evidently, this change leads to a higher battery 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 counterpart are plotted in Fig. 6.

For the next anomaly type, we reduce the requested motor torque value output by the powertrain control module by 10 %. As a response to the change, 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 counterpart are plotted in Fig. 7.

In the next case, we increase the loaded wheel diameter by 10 % which, for the same target vehicle speed, leads to a lower motor and axle angular speed. Furthermore, a larger wheel diameter leads to higher motor and axle torques, which are achieved using higher accelerator and brake pedal values. Here, the wheel diameter also has an effect on the battery temperature, which, depending on its absolute magnitude, may also affect the cooling system. Due to model limitations, this anomaly can only be simulated for whole-sequence anomalies. One of the anomalous time series for the HUDDS drive cycle and its control counterpart are plotted in Fig. 8.

The last anomaly is recorded after 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 battery 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 counterpart are plotted in Fig. 9.

To ensure that the different anomaly types actually lead to anomalous behaviour, we run control simulations with the same initial battery

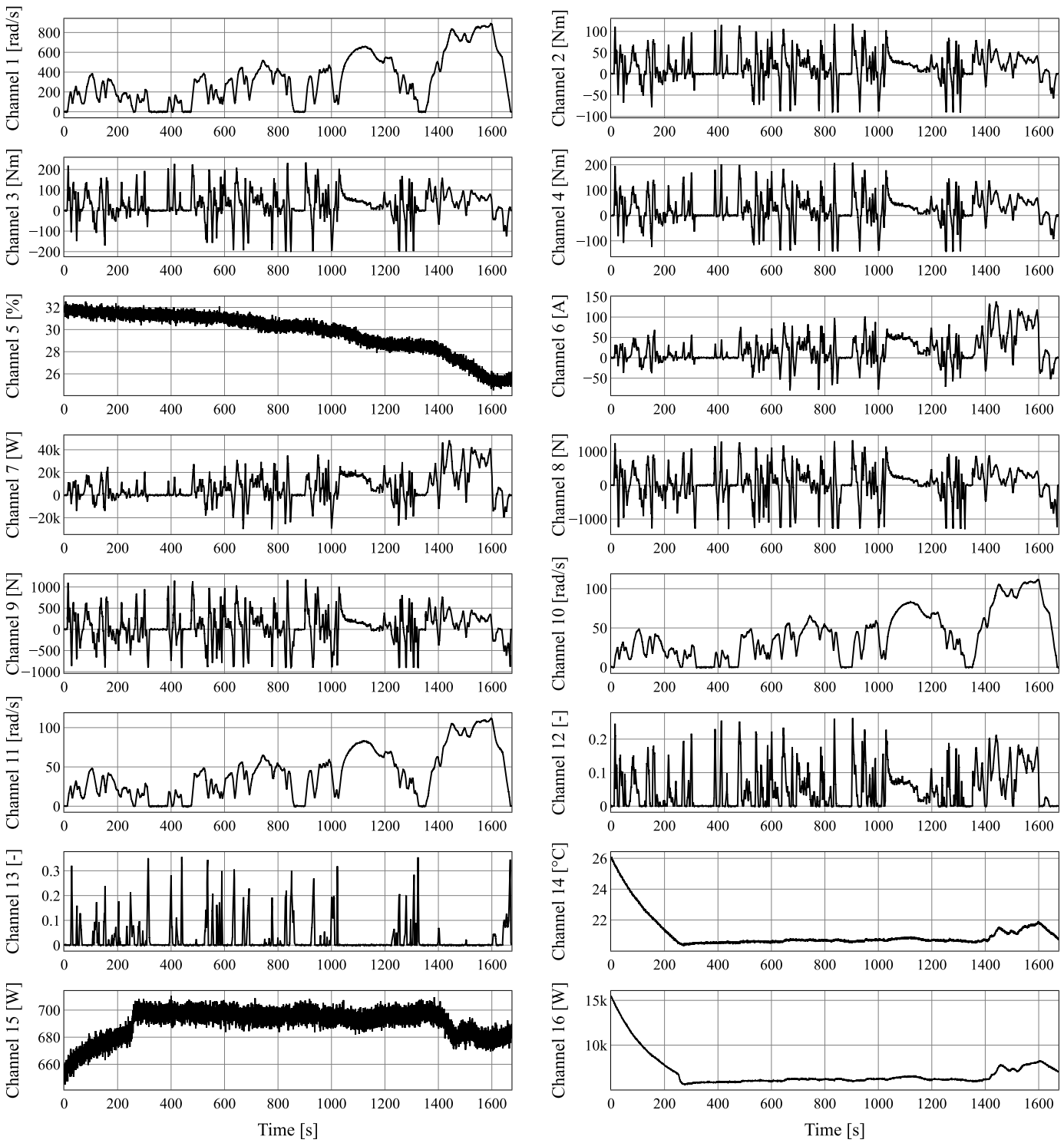

Fig. 3. Sample plot of a *nominal* sequence with added noise and undergone trimming. The channel legend can be found in Table 2.

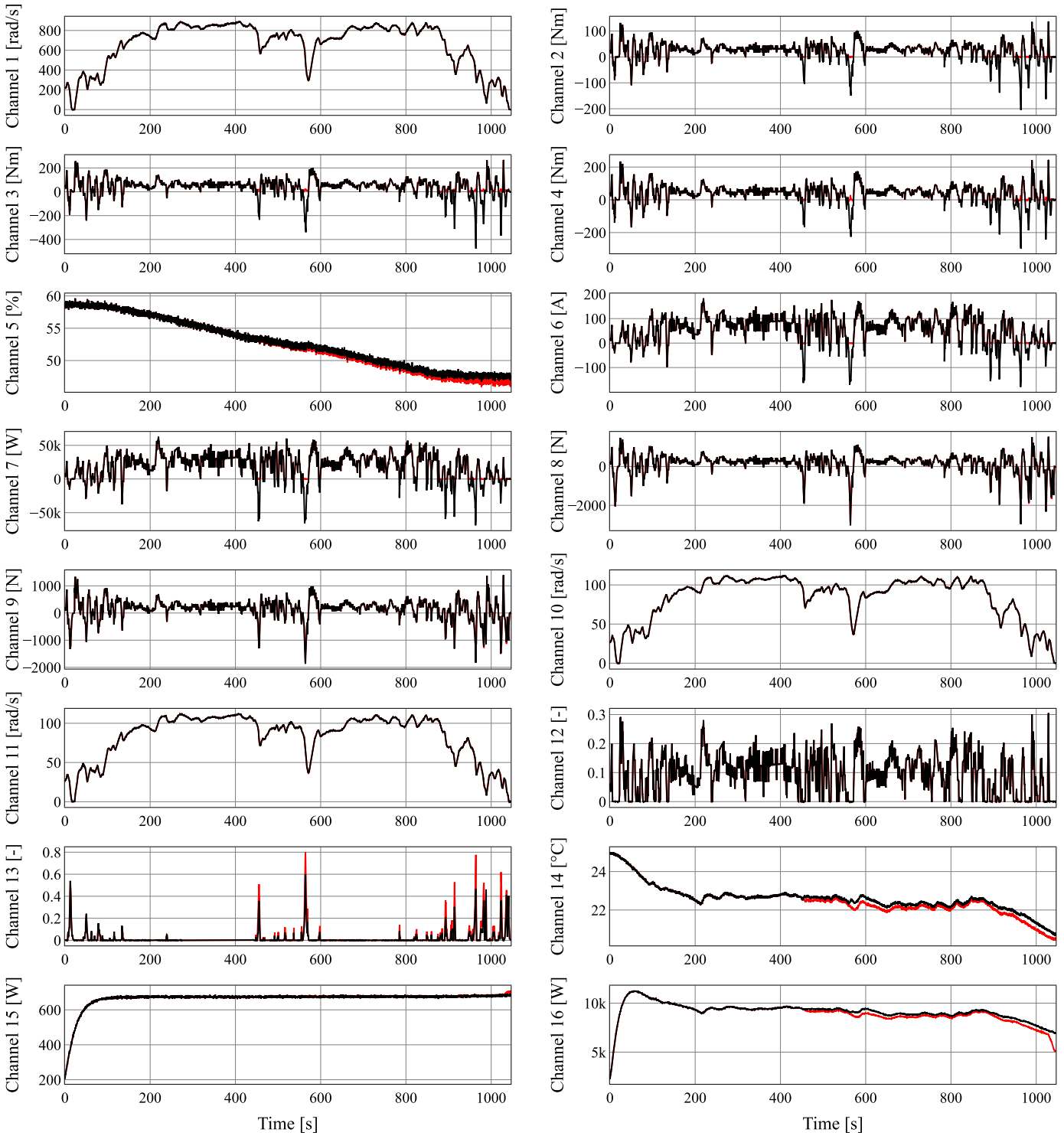

Fig. 4. Plot of an *anomalous* sequence without regenerative braking (in red) and its control counterpart (in black), both with added noise and undergone trimming. The anomalous sub-sequence starts after 384.6 s. The channel legend can be found in Table 2.

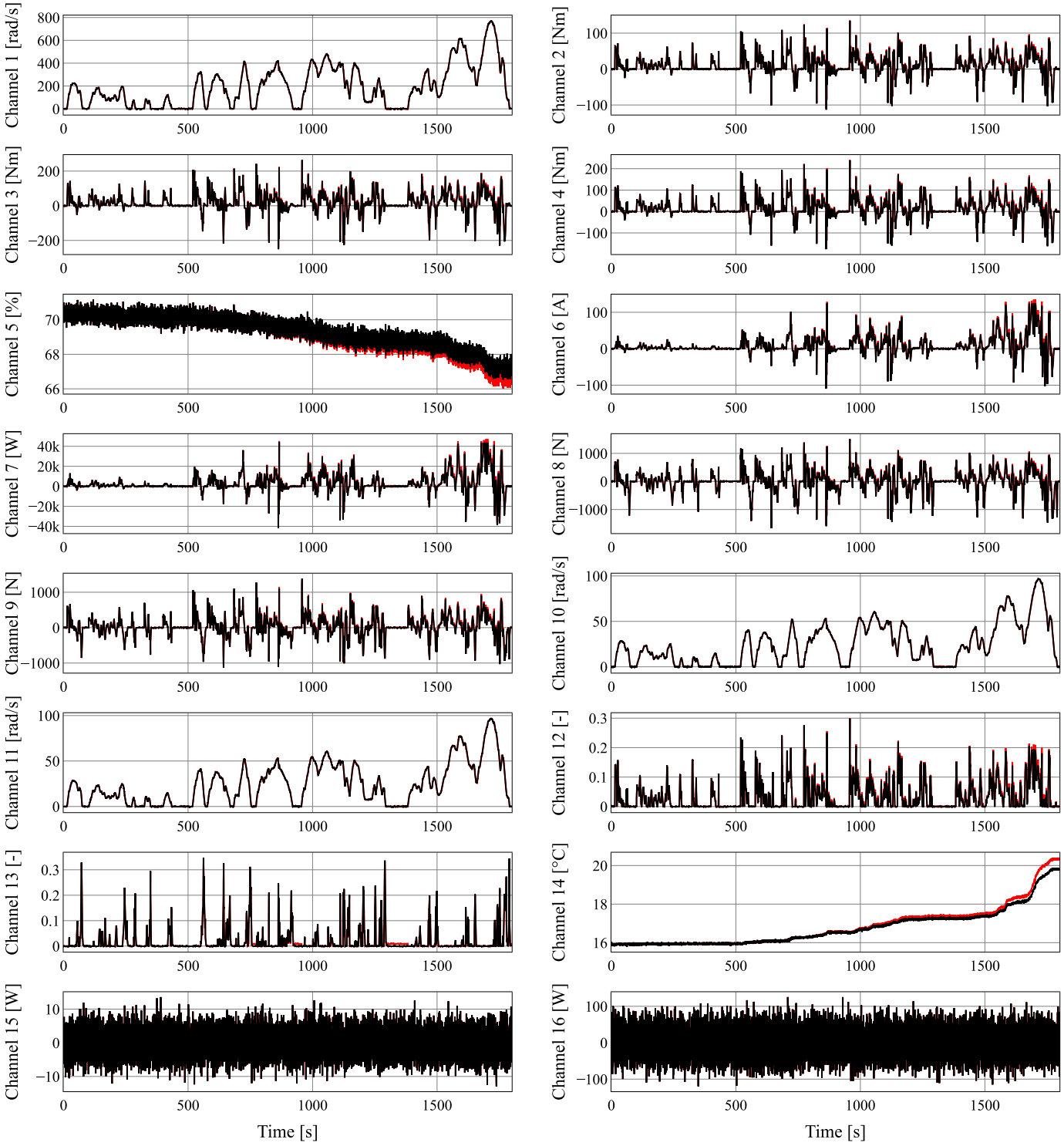

Fig. 5. Plot of an *anomalous* sequence with an added headwind (in red) and its control counterpart (in black), both with added noise and undergone trimming. The anomalous sub-sequence starts after 738.0 s. The channel legend can be found in Table 2.

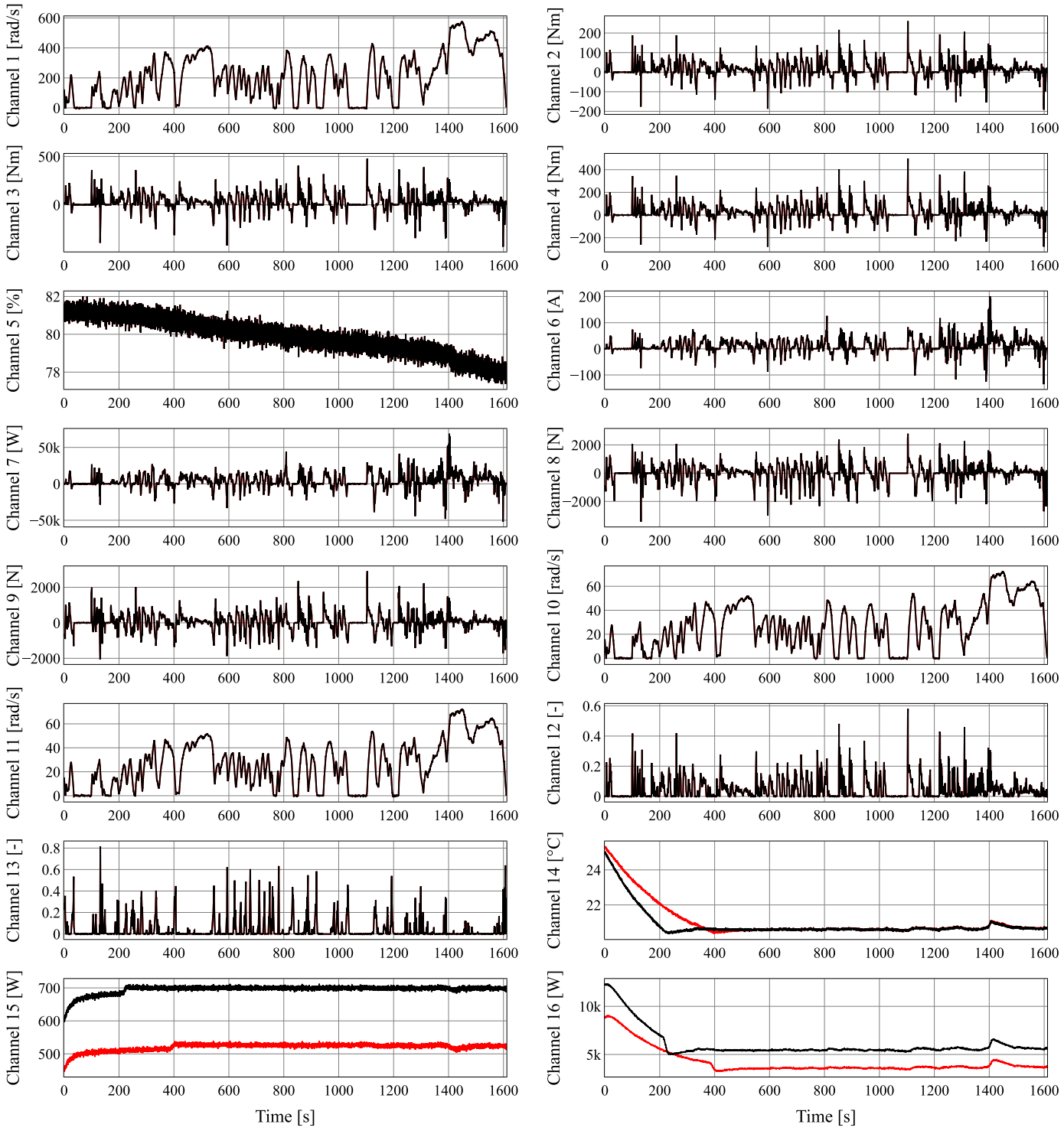

Fig. 6. Plot of an *anomalous* sequence with a reduced cooling pump displacement (in red) and its control counterpart (in black), both with added noise and undergone trimming. It is a whole-sequence anomaly, and hence the anomalous behaviour starts from the first time step. The channel legend can be found in Table 2.

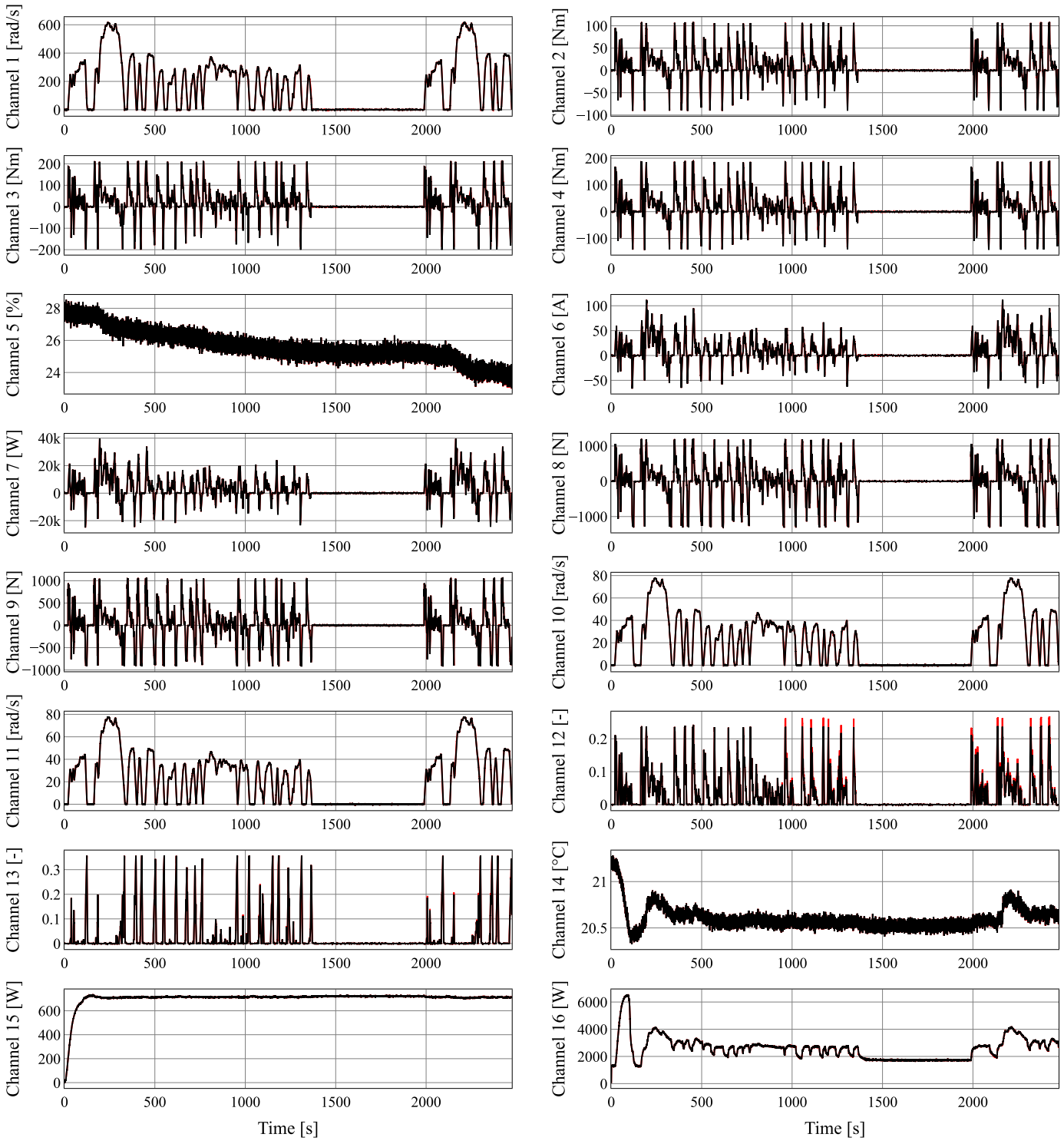

Fig. 7. Plot of an *anomalous* sequence with a reduced requested motor torque (in red) and its control counterpart (in black), both with added noise and undergone trimming. The anomalous sub-sequence starts after 940.2 s. The channel legend can be found in Table 2.

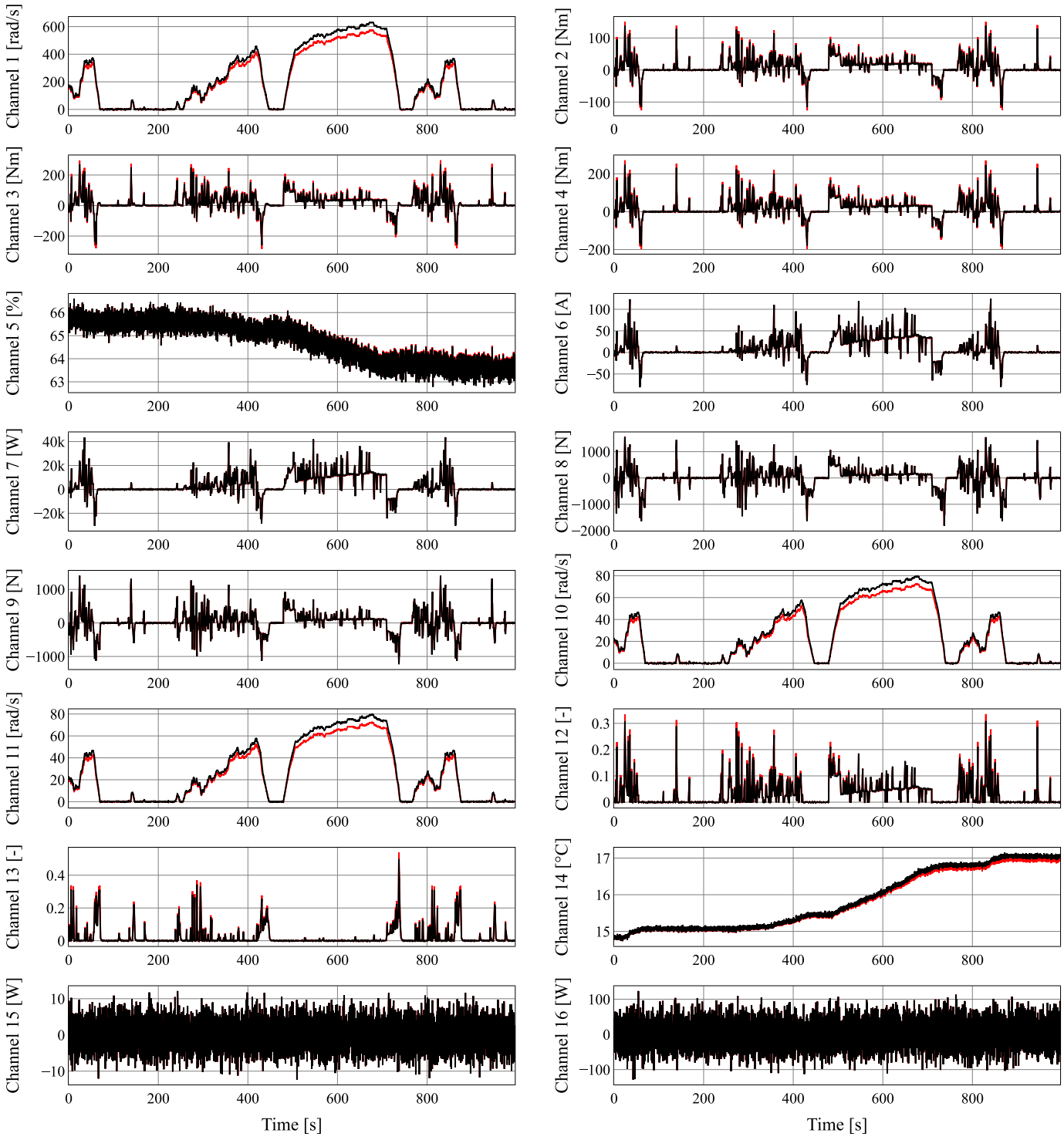

Fig. 8. Plot of an *anomalous* sequence with an increased loaded wheel diameter (in red) and its control counterpart (in black), both with added noise and undergone trimming. It is a whole-sequence anomaly, and hence the anomalous behaviour starts from the first time step. The channel legend can be found in Table 2.

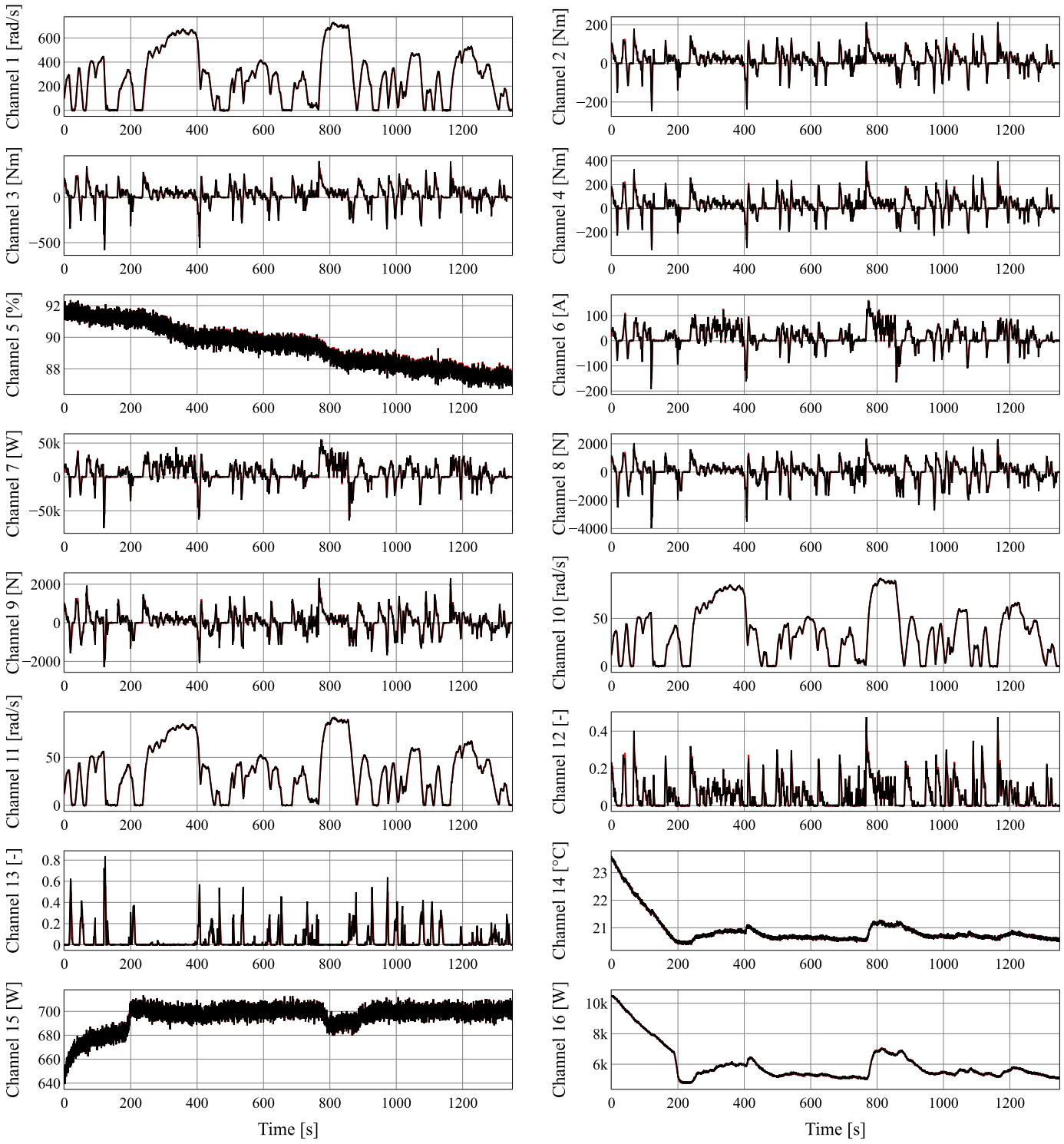

Fig. 9. Plot of an *anomalous* sequence with an increased driver response time (in red) and its control counterpart (in black), both with added noise and undergone trimming. It is a whole-sequence anomaly, and hence the anomalous behaviour starts from the first time step. The channel legend can be found in Table 2.

states but with otherwise nominal model properties. Given the uniform distribution from which the battery temperature is sampled from, half of the simulated anomaly types start with a battery temperature below 20 °C. In these cases, the battery will naturally heat up as it is being used and hence the cooling system does not play a role. 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 $N_a = 284$ successful anomalous simulations, where $N_a = N_{ss} + N_{ws}$. Hence, the entire dataset $\mathcal{D}$ consists of $N_n + N_{ss} + N_{ws} = 3557$ unique (nominal and anomalous) multivariate time series, with an anomalous sequence ratio of $284/3557 \approx 8\%$. $\mathcal{D}$ is then shuffled and divided into three separate folds for cross-validation, which corresponds to 2/3 and 1/3 split training and test subsets, respectively. Formally, the training subset $\mathcal{D}^{\text{train}} = \{S_1, ..., S_m, ..., S_M\}$ then consists of $M = 2371$ multivariate time series on average, where $S_m \in \mathbb{R}^{T_m \times d_\mathcal{D}}$, where T_m is the number of time steps in sequence S_m. Likewise, the test subset $\mathcal{D}^{\text{test}} = \{S_1, ..., S_n, ..., S_N\}$ consists of $N = 1186$ multivariate time series on average, where $S_n \in \mathbb{R}^{T_n \times d_\mathcal{D}}$, where T_n is the number of time steps in sequence S_n. For benchmarking purposes, we suggest the users use the prescribed training and test split to ensure comparable results.

To add further complexity and to reflect certain real-world artefacts, we undertake some *post-processing*. First, we trim the beginning of each time series in $\mathcal{D}$ by random amounts 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, we add noise sampled from Gaussian distribution $\mathcal{N}(0, 0.01\sigma_\mathcal{D})$ to further move the obtained data towards the real world, where $\sigma_\mathcal{D}$ is the feature-wise standard deviation of the dataset.

3.3. Usability of the dataset

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

We believe the underlying properties of the dataset can be useful in other research areas too. The same $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$ subsets can also be used for *imbalanced time series classification* if the labels are considered. Additionally, we provide a number of different subset variations for other tasks. For *semi-supervised* anomaly detection, we provide a clean $\mathcal{D}^{\text{train}}$ with on average $M = 2182$ nominal time series and the same labelled $\mathcal{D}^{\text{test}}$ in the dataset, where clean refers to the absence of anomalous sequences in the subset. Furthermore, for time series *forecasting* or *generation*, we supply clean versions of both $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$, where $M = 2182$ and $N = 1091$ on average, respectively. Despite being targeted at *online* time series anomaly detection, the PATH dataset can just as well be used in offline time series anomaly detection.

4. Baseline results on the dataset

4.1. Methodology

The evaluation metrics used to quantify anomaly detection performance by Correia et al. [19] are adopted, as they provide a parameter-free way to quantify *online* anomaly detection performance in an interpretable way. Said metrics are very similar to the conventional true positive, false positive, true negative and false negative labels applied to each individual discrete sequence, with the exception that a sub-sequence anomaly can also be labelled as a false positive if detected too early. In addition to that, the time between detection and ground-truth anomaly start is also quantified for each anomalous 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 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. [15], 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 Fig. 10. For a very low threshold τ_1, the anomaly is considered a false 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, 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 uncallibrated detection performance. Despite its short-comings, it is the only set of metrics that are apt for online discrete-sequence problems.

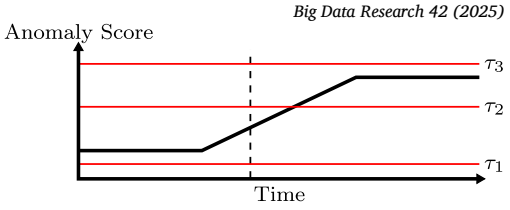

Fig. 10. Arbitrary anomaly score as a function of time. The dotted vertical line denotes the start of the anomalous behaviour and the red lines represent three different thresholds.

In this work, we consider VS-VAE [32], OmniAnomaly [2], VASP [23], TCN-AE [24], SISVAE [25], LW-VAE [26], TSADIS [27], and TeVAE [19] when conducting experiments. The hyperparameters for each approach are set as specified in the respective publication, 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.

The validation subset $\mathcal{D}^{\text{val}}$ is obtained by further splitting $\mathcal{D}^{\text{train}}$ and hence is also unlabelled. As future work may not require a validation subset, it is left to the individual to extract it from the training subset if needed. The test subset $\mathcal{D}^{\text{test}}$ should be the same as the one provided to ensure comparable results.

As mentioned in Section 3 the simulation signals are sampled at 10 Hz by default, however, to reduce the computational load in our experiments, we downsample the data to 2 Hz with a low-pass filter with a cut-off frequency of 1 Hz, as it is consistent with the Whittaker–Nyquist–Shannon theorem [28]. This downsampling procedure is considered as part of the approaches tested and is optional for any future work, which may alternatively use the raw time series data or perhaps even correlation matrices [29,30].

To bring all channels to a common magnitude, the dataset features are z-score normalised.

Finally, we segment the time series data into fixed-length sub-sequences, also referred to as *windows*. The rationale for using windows instead of full-length sequences is that the dynamics present in the time series data tend to occur quickly and only influence the data for a brief duration. Modelling entire variable-length sequences is possible, but it would lead to inefficient use of the model's learning capacity, as it would have to maintain information over unnecessarily long periods. By focusing on windows that are just long enough to capture the existing dynamics in the data, model training should be more effective. To determine the optimal window length at 2 Hz, especially to capture the slowest dynamics present in a signal, we perform an autocorrelation analysis [19] for each drive cycle and for every feature within those cycles, yielding a window size of 256 time steps. In the literature, the

window size is often treated as a hyperparameter [26,31,27] or provided without reasoning [32,33,25,24,34]. However, it is not possible to tune hyperparameters outside a supervised setting, and therefore such methods might not be applicable in real-world settings. In contrast, finding a suitable window length using autocorrelation is completely unsupervised. TSADIS takes window size as a hyperparameter before calling, hence a window size of 256 time steps is also used. To map the individual windows back to continuous sequences, mean-type reverse-windowing [19] is used where applicable.

4.2. Reproducibility and benchmarking considerations

While perhaps sounding similar, repeatability, reproducibility, and replicability are defined differently according to the Association for Computing Machinery [35].

- Repeatability: the property of the research's finding being obtainable using the same experimental setup by the same person.
- Reproducibility: the property of the research's finding being obtainable using the same experimental setup by a *different* person.
- Replicability: the property of the research's finding being obtainable using the *different* experimental setup by a *different* person.

Several position papers [36–42] call for greater attention to be paid to reproducibility and replicability in computer science. Additionally, some conferences focus on reproduction, like the Machine Learning Reproducibility Challenge [43], or make specific calls for reproducibility and replicability papers, like the European Conference on Information Retrieval [44]. To enable future work to reproduce the results in this paper, we aim to be as transparent as possible by providing publicly available, clean and thoroughly commented source code for all experiments and the Simulink model under https://github.com/lcs-crr/PATH, as is suggested in literature [37,38,42].

The seed for random operations has an impact on model training, given that processes like sampling and weight initialisation rely on it. To increase robustness of the results and to eliminate the possibility of the results owing to a specific fold and seed combination rather from the characteristics of the model [42], all three folds are trained on seeds 1 through 3, yielding 9 different combinations. The final result is then given as the average of the 9 different combinations. As mentioned, TSADIS does not require training, and hence its results are simply the average over all three folds.

In case future work aims to replicate the results of this paper, we encourage deviating from the experimental setup outlined in this paper [45], though, as Bartz-Beielstein et al. [40] point out, there is no definition for how different an experimental setup needs to be for results to be considered replicable. Using a different set of seeds, splitting the dataset into different folds, using different implementations of the approaches or even by using different software and hardware are some of the variables that could be changed in the setup, for example. In the case of replicability, these changes should not change the outcome [39]. Moreover, it is just as important that future work provides the same level of transparency regarding the experimental setup and documentation.

It should be noted that the test subset $\mathcal{D}^{\text{test}}$ is often not available in the real world, so we strongly discourage approaches performing supervised threshold search using the labelled test data in $\mathcal{D}^{\text{test}}$.

Furthermore, there is no way to stop future research from performing hyperparameter tuning using $\mathcal{D}^{\text{test}}$, hence any results obtained for this dataset should be considered as the theoretical maximum anomaly detection performance achievable by the approach, not as a realistic anomaly detection performance observable in the real world.

We run all simulations that generate the PATH dataset on a workstation equipped with an Intel Xeon Gold 6234 CPU running Windows 10 Enterprise LTSC version 21H2 with MATLAB 2023b. 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 repository.

Table 5
Mean ± standard deviation of F_1 score, precision P, recall R, and average detection delay $\bar{\delta}$ using the unsupervised threshold (top half) and theoretical best threshold (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 given in **bold**.

Approach	F_1	P	R	$\bar{\delta}$ [s]
VS-VAE	0.03 ± 0.02	0.78 ± 0.41	0.01 ± 0.01	991.4 ± 71.1
OmniA	$\mathbf{0.06 \pm 0.06}$	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
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	1209.8 ± 8.9
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
OmniA	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
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	1209.8 ± 8.9
TeVAE	$\mathbf{0.58 \pm 0.08}$	0.69 ± 0.12	0.50 ± 0.09	676.4 ± 103.2

4.3. Results and discussion

To provide baseline results for the version of the PATH dataset for *unsupervised* anomaly detection, we test several approaches. The corresponding results are shown in Table 5.

First, it is evident that there is a large gap between the unsupervised and theoretical best threshold results. The unsupervised threshold is a rudimentary estimation based on the unlabelled validation subset $\mathcal{D}^{\text{val}}$ [19] and tends to be set higher than the theoretical best. This is because in the unsupervised version of the dataset, there are anomalous sequences within $\mathcal{D}^{\text{val}}$, which are associated with a higher maximum anomaly score and therefore threshold. It is clear, however, that the results are far from good, which sets one foundation for future work. We can isolate absolute detection performance from the threshold choice by performing a grid search on different thresholds, which allows us to find the threshold yielding the theoretical best F_1 score. At this threshold, we observe the best possible anomaly detection performance the approach can achieve on the test set, though it is not observable in the real-world. Here, TeVAE performs best in terms of F_1 score, though with a high average detection delay due to the number of high number of false negatives. While these results are better than with the unsupervised threshold, they still leave room for improvement. Regardless of the results, it cannot be denied how much less computationally intensive TSADIS is compared to methods based on deep learning. Even on commodity hardware, more specifically a laptop with an Intel Core i7-1185G7, it can evaluate test data faster than deep learning models. It should be noted that this is mainly because no reverse-windowing is needed with TSADIS, a process that, unlike inference in deep learning models, runs on the CPU, not the GPU. However, because TSADIS works on the entire sequence, not on windows, it is technically an offline approach, hence why its detection delay is the highest. Additionally, TSADIS does *not* require a training procedure. At first glance, this property is a benefit, as the implementation hurdle is much lower than with deep learning models, which essentially require GPU-acceleration. However, without training data, TSADIS cannot know what is a nominal time series and what is not, therefore the nominal behaviour is *not modelled*. It can solely rely on the information present within a sequence to calculate an anomaly

Table 6
Mean ± standard deviation of F_1 score, precision P, recall R, and average detection delay $\bar{\delta}$ using the unsupervised threshold (top half) and theoretical best threshold (bottom half) for TeVAE applied to the *semi-supervised* anomaly detection version of the PATH dataset, i.e. the version without anomalous data in training subset $\mathcal{D}^{\text{train}}$.

Approach	F_1	P	R	$\bar{\delta}$ [s]
TeVAE	0.63 ± 0.22	0.95 ± 0.06	0.52 ± 0.26	589.1 ± 212.0
TeVAE	0.80 ± 0.15	0.88 ± 0.11	0.76 ± 0.18	412.6 ± 143.8

score, which is part of the reason it cannot outperform deep learning-based methods.

We also performed limited testing on the version of the dataset for *semi-supervised* anomaly detection. It involves the same testing procedure, except that the clean version, i.e. anomaly-free, of the training subset is used.

The corresponding results for TeVAE are shown in Table 6. The gap between results obtained using the unsupervised threshold and the theoretical best is now much smaller than observed in Table 5, which can be attributed to the lack of anomalous data in $\mathcal{D}^{\text{val}}$. Additionally, there is a large gap between the theoretical best results between the unsupervised and semi-supervised versions, indicating the need for more robust future work when anomalous data is present in $\mathcal{D}^{\text{train}}$.

5. Conclusion and outlook

We propose a novel multivariate time series dataset for online anomaly detection, called the Powertrain Anomaly Time series benchMark (PATH) dataset. The PATH dataset is generated using simulation, where the model it is based on resembles a real-world dynamic system. In addition to that, simulation is done in a variety of different initial states to further add to the diversity of the dataset. To increase the complexity of the dataset, noise is applied and the beginning of time series are randomly trimmed. The anomalies in the PATH dataset arise from changing parameters prior to simulation, as opposed to manual data manipulation, resulting in anomalies that are non-trivial and realistic. We offer the dataset in three different versions: one for unsupervised anomaly detection, where the training subset consists of both anomalous and nominal sequences, another for semi-supervised anomaly detection, where the training subset consists of nominal sequences only, and one for time series generation or forecasting, where both the training and test subsets are nominal. Lastly, for each of the versions, we offer three different folds with a pre-defined train and test split to ensure generalised and comparable results.

The experiments conducted in this work further support the claim of non-triviality because, even when the threshold choice is removed as a factor, the best approach in an unsupervised setting only manages to achieve an F_1 score of 0.58 and an average detection delay of 676.4 s. In contrast, however, the results significantly improve when the clean version of the test subset is used. Here, the average theoretical best F_1 score reaches 0.80 and an average detection delay of 412.6 s, highlighting the need for methods more robust to anomalous data in the unlabelled and contaminated training subset.

In the future, the PATH dataset should be extended to a standardised benchmark consisting of not only a dataset based on the longitudinal electric vehicle dynamics, but also on simulation models from other domains. Additionally, the simulation model can be adapted to take other factors into account, like battery ageing. Battery ageing can be characterised by the charge capacity, which, fixed in this dataset, can be changed dynamically to simulate battery ageing, which will have an impact on the entire system. This property can be especially useful for research in the area of unsupervised predictive maintenance, where an explicit health signal is not present. Additionally, modal channels could be considered if a gearbox model was implemented, since, depending on the discrete gear, the motor speed is decoupled from the axle speed. Furthermore, there is a need for more sophisticated evolution of the online evaluation metrics [19] that do not assume a single anomalous subsequence per test time series and that are recall consistent. When said metrics are available, the dataset could be extended to such anomalies types.

CRediT authorship contribution statement

Lucas Correia: Conceptualization, Visualization, Formal analysis, Writing – review & editing, Validation, Data curation, Writing – original draft, Methodology, Investigation. **Jan-Christoph Goos:** Supervision, Writing – review & editing. **Thomas Bäck:** Writing – review & editing, Supervision. **Anna V. Kononova:** Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The research data and source code are openly available from Zenodo and GitHub, respectively.

References

[1] G. Moody, R. Mark, The impact of the MIT-BIH arrhythmia database, IEEE Engineering in Medicine and Biology 20 (2001) 45–50, https://doi.org/10.13026/C2F305.
[2] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, D. Pei, Robust anomaly detection for multivariate time series through stochastic recurrent neural network, in: International Conference on Knowledge Discovery & Data Mining (KDD), 2019.
[3] A.P. Mathur, N.O. Tippenhauer, SWaT: a water treatment testbed for research and training on ICS security, in: International Workshop on Cyber-Physical Systems for Smart Water Networks (CySWater), IEEE, 2016, pp. 31–36.
[4] C.M. Ahmed, V.R. Palleti, A.P. Mathur, WADI: a water distribution testbed for research in the design of secure cyber physical systems, in: 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks, CySWATER 2017, 2017, pp. 25–28.
[5] R. Zhang, P. Zhou, J. Qiao, Anomaly detection of nonstationary long-memory processes based on fractional cointegration vector autoregression, IEEE Transactions on Reliability 72 (2023) 1383–1394, https://doi.org/10.1109/TR.2023.3314429.
[6] C. Bachechi, L. Po, F. Rollo, Big data analytics and visualization in traffic monitoring, Big Data Research 27 (2022) 100292, https://doi.org/10.1016/j.bdr.2021.100292.
[7] L. Zhao, B. Guo, C. Dai, Y. Shen, F. Chen, M. Zhao, Y. Hu, Multi-step trend aware graph neural network for traffic flow forecasting, Big Data Research 38 (2024) 100482, https://doi.org/10.1016/j.bdr.2024.100482.
[8] Z. Zhao, G. Shen, L. Wang, X. Kong, Graph spatial-temporal transformer network for traffic prediction, Big Data Research 36 (2024) 100427, https://doi.org/10.1016/j.bdr.2024.100427.
[9] L. Correia, J.-C. Goos, P. Klein, T. Bäck, A.V. Kononova, Online model-based anomaly detection in multivariate time series: taxonomy, survey, research challenges and future directions, Engineering Applications of Artificial Intelligence 138 (2024) 109323, https://doi.org/10.1016/j.engappai.2024.109323.
[10] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection for discrete sequences: a survey, IEEE Transactions on Knowledge and Data Engineering 24 (2012) 823–839, https://doi.org/10.1109/TKDE.2010.235.
[11] S. Schmidl, P. Wenig, T. Papenbrock, Anomaly detection in time series: a comprehensive evaluation, Proceedings of the VLDB Endowment 15 (2022) 1779–1797, https://doi.org/10.14778/3538598.3538602.
[12] L. Correia, J.-C. Goos, T. Bäck, A.V. Kononova, Path: a discrete-sequence dataset for evaluating online unsupervised anomaly detection approaches for multivariate time series, https://doi.org/10.5281/zenodo.14892756, 2025.
[13] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, T. Soderstrom, Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding, in: ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM, 2018, pp. 387–395.
[14] R. Wu, E.J. Keogh, Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress, IEEE Transactions on Knowledge and Data Engineering (2021) 2421–2429, https://doi.org/10.1109/TKDE.2021.3112126.
[15] D. Wagner, T. Michels, F.C.F. Schulz, A. Nair, M. Rudolph, M. Kloft, TimeSeAD: benchmarking deep multivariate time-series anomaly detection, Transactions on Machine Learning Research (2023).

[16] P. Wenig, S. Schmidl, T. Papenbrock, Anomaly detectors for multivariate time series: the proof of the pudding is in the eating, in: 2024 IEEE 40th International Conference on Data Engineering Workshops (ICDEW), IEEE, Utrecht, Netherlands, 2024, pp. 96–101.

[17] P. Wenig, S. Schmidl, T. Papenbrock, TimeEval: a benchmarking toolkit for time series anomaly detection algorithms, Proceedings of the VLDB Endowment 15 (2022) 3678–3681, https://doi.org/10.14778/3554821.3554873.

[18] D. Baumgartner, H. Langseth, H. Ramampiaro, K. Engø-Monsen, mTADS: multivariate time series anomaly detection benchmark suites, in: 2023 IEEE International Conference on Big Data (BigData), IEEE, Sorrento, Italy, 2023, pp. 588–597.

[19] L. Correia, J.-C. Goos, P. Klein, T. Bäck, A.V. Kononova, TeVAE: a variational autoencoder approach for discrete online anomaly detection in variable-state multivariate time-series data, in: Computational Intelligence, vol. 1196, Springer Nature Switzerland, Cham, 2025, pp. 106–132.

[20] J. Audibert, P. Michiardi, F. Guyard, S. Marti, M.A. Zuluaga, Do deep neural networks contribute to multivariate time series anomaly detection?, Pattern Recognition 132 (2022) 108945, https://doi.org/10.1016/j.patcog.2022.108945.

[21] F. Rewicki, J. Denzler, J. Niebling, Is it worth it? Comparing six deep and classical methods for unsupervised anomaly detection in time series, Applied Sciences 13 (2023) 1778, https://doi.org/10.3390/app13031778.

[22] H.A. Dau, A. Bagnall, K. Kamgar, C.-C.M. Yeh, Y. Zhu, S. Gharghabi, C.A. Ratanamahatana, E. Keogh, The UCR time series archive, IEEE/CAA Journal of Automatica Sinica 6 (2019) 1293–1305, https://doi.org/10.1109/JAS.2019.1911747.

[23] J. von Schleinitz, M. Graf, W. Trutschnig, A. Schröder, VASP: an autoencoder-based approach for multivariate anomaly detection and robust time series prediction with application in motorsport, Engineering Applications of Artificial Intelligence 104 (2021) 104354, https://doi.org/10.1016/j.engappai.2021.104354.

[24] M. Thill, W. Konen, H. Wang, T. Bäck, Temporal convolutional autoencoder for unsupervised anomaly detection in time series, Applied Soft Computing 112 (2021) 107751, https://doi.org/10.1016/j.asoc.2021.107751.

[25] L. Li, J. Yan, H. Wang, Y. Jin, Anomaly detection of time series with smoothness-inducing sequential variational auto-encoder, Transactions on Neural Networks and Learning Systems 32 (2021) 1177–1191, https://doi.org/10.1109/TNNLS.2020.2980749.

[26] D. Fährmann, N. Damer, F. Kirchbuchner, A. Kuijper, Lightweight long short-term memory variational auto-encoder for multivariate time series anomaly detection in industrial control systems, Sensors 22 (2022) 2886, https://doi.org/10.3390/s22082886.

[27] S. Tafazoli, E. Keogh, Matrix profile XXVIII: discovering multi-dimensional time series anomalies with K of N anomaly detection, in: International Conference on Data Mining (SDM), Society for Industrial and Applied Mathematics, 2023.

[28] C. Shannon, Communication in the presence of noise, Proceedings of the IRE 37 (1949) 10–21, https://doi.org/10.1109/JRPROC.1949.232969.

[29] T. Tayeh, S. Aburakhia, R. Myers, A. Shami, An attention-based ConvLSTM autoencoder with dynamic thresholding for unsupervised anomaly detection in multivariate time series, Machine Learning and Knowledge Extraction 4 (2022) 350–370, https://doi.org/10.3390/make4020015.

[30] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, N.V. Chawla, A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data, in: AAAI Conference on Artificial Intelligence, 2019.

[31] S. Tuli, G. Casale, N.R. Jennings, TranAD: deep transformer networks for anomaly detection in multivariate time series data, in: Very Large Databases (VLDB), 2022.

[32] J. Pereira, M. Silveira, Unsupervised anomaly detection in energy time series data using variational recurrent autoencoders with attention, in: International Conference on Machine Learning and Applications (ICMLA), 2018.

[33] T. Chen, X. Liu, B. Xia, W. Wang, Y. Lai, Unsupervised anomaly detection of industrial robots using sliding-window convolutional variational autoencoder, IEEE Access 8 (2020) 47072–47081, https://doi.org/10.1109/ACCESS.2020.2977892.

[34] K. Doshi, S. Abudalou, Y. Yilmaz, Reward once, penalize once: rectifying time series anomaly detection, in: International Joint Conference on Neural Networks (IJCNN), 2022.

[35] Artifact review and badging - current, https://www.acm.org/publications/policies/artifact-review-and-badging-current, 2020.

[36] C. Drummond, Replicability is not reproducibility: nor is it good science, in: Evaluation Methods for Machine Learning Workshop at the 26th ICML, 2009.

[37] R.D. Peng, Reproducible research in computational science, Science 334 (2011) 1226–1227, https://doi.org/10.1126/science.1213847.

[38] M.R. Munafò, B.A. Nosek, D.V.M. Bishop, K.S. Button, C.D. Chambers, N. Percie Du Sert, U. Simonsohn, E.-J. Wagenmakers, J.J. Ware, J.P.A. Ioannidis, A manifesto for reproducible science, Nature Human Behaviour 1 (2017) 0021, https://doi.org/10.1038/s41562-016-0021.

[39] O.E. Gundersen, S. Kjensmo, State of the art: reproducibility in artificial intelligence, in: AAAI Conference on Artificial Intelligence, vol. 32, 2018.

[40] T. Bartz-Beielstein, C. Doerr, D.v.d. Berg, J. Bossek, S. Chandrasekaran, T. Eftimov, A. Fischbach, P. Kerschke, W.L. Cava, M. Lopez-Ibanez, K.M. Malan, J.H. Moore, B. Naujoks, P. Orzechowski, V. Volz, M. Wagner, T. Weise, Benchmarking in optimization: best practice and open issues, arXiv:2007.03488 [cs], 2020.

[41] S. Kapoor, A. Narayanan, Leakage and the reproducibility crisis in machine-learning-based science, Patterns 4 (2023) 100804, https://doi.org/10.1016/j.patter.2023.100804.

[42] H. Semmelrock, S. Kopeinik, D. Theiler, T. Ross-Hellauer, D. Kowald, Reproducibility in machine learning-driven research, arXiv:2307.10320 [cs], 2023.

[43] MLRC, ML Reproducibility Challenge 2023, 2023, https://reproml.org/.

[44] ECIR, Call for Reproducibility Papers, 2025, https://ecir2025.eu/call-for-reproducibility-papers/.

[45] S.N. Goodman, D. Fanelli, J.P.A. Ioannidis, What does research reproducibility mean?, Science Translational Medicine 8 (2016), https://doi.org/10.1126/scitranslmed.aaf5027.