Ahrens *et al. EURASIP Journal on Advances in Signal Processing* (2019) 2019:27
https://doi.org/10.1186/s13634-019-0619-3

EURASIP Journal on Advances in Signal Processing

RESEARCH Open Access

A machine-learning phase classification scheme for anomaly detection in signals with periodic characteristics

Lia Ahrens[1]*, Julian Ahrens[1] and Hans D. Schotten[1,2]

Abstract

In this paper, we propose a novel machine-learning method for anomaly detection applicable to data with periodic characteristics where randomly varying period lengths are explicitly allowed. A multi-dimensional time series analysis is conducted by training a data-adapted classifier consisting of deep convolutional neural networks performing phase classification. The entire algorithm including data pre-processing, period detection, segmentation, and even dynamic adjustment of the neural networks is implemented for fully automatic execution. The proposed method is evaluated on three example datasets from the areas of cardiology, intrusion detection, and signal processing, presenting reasonable performance.

Keywords: Anomaly detection, Time series analysis, Phase classification, Machine learning, Convolutional neural networks

1 Introduction

Many real-world systems, both natural and anthropogenic, exhibit periodic behaviour. Monitoring such systems necessarily produces periodic time series. In one particular instance of such a monitoring application, one is interested in automatically detecting changes in the periodically repeating pattern and thus anomalies in the systems operation. This type of anomaly detection occurs in a wide range of different fields and applications, be they medical, e.g. diagnosing diseases of the cardiovascular and respiratory systems, in industrial contexts, e.g. monitoring the operation of a transformer or rotating machinery, and in signal processing and communications. The pursued aims range from simple monitoring to intrusion detection and prevention.

Traditionally, anomaly detection is performed in the form of outlier detection in mathematical statistics. Numerous methods have been proposed, including but not limited to distance- and density-based techniques [1, 2] and subspace- or submanifold-based techniques [3–5]. Most of these approaches make no explicit use of the concept of time and are therefore usually less suited for the analysis of time series. Methods making explicit use of the temporal structure include classical models from statistical time series analysis such as autoregressive–moving average (ARMA) models [6], Kalman filters [7] or more general hidden Markov models [8], and rolling-window distance-based methods such as matrix profiles [9]. Distance analysing methods are effective for clean data but not robust against noise, whereas distribution-based methods from mathematical statistics are still powerful in the presence of noise, requiring data-specific parameterisation. In the past few years, non-linear methods, such as different types of recurrent neural networks (RNNs) and in particular long short-term memory (LSTM) networks have also come into use [10, 11]. Many of these methods are difficult to train [12–14] or need large amounts of data in order to achieve reasonable performance while avoiding overfitting. On the other hand, in recent years, convolutional neural networks (CNNs) have gained popularity in image processing [15, 16] where they are used mainly for classification tasks. The same principles that are responsible for the success of CNNs in image processing carry over to other types of signal processing when the number of dimensions of the convolutional kernels is changed accordingly. Most of the work using recurrent or

*Correspondence: Lia.Ahrens@dfki.de
[1] Deutsches Forschungszentrum für Künstliche Intelligenz, Trippstadter Straße 122, 67663 Kaiserslautern, Germany
Full list of author information is available at the end of the article

convolutional networks for time series analysis focuses on forecasting or detecting certain patterns explicitly known at training time. On these tasks, convolutional networks have recently been shown to outperform the previously state of the art LSTMs [17].

In this paper, we consider data with periodic characteristics and design a machine-learning algorithm for time series analysis, in particular anomaly detection, applying convolutional neural nets in a manner which, to the best of the authors' knowledge, has not been proposed previously. In contrast to existing methods and inspired by machine-learning methods for image processing, we employ a convolutional net acting not as a predictor or estimator but as a classifier whose classes indicate phase, i.e. the relative location in time. We also integrate general procedures for data pre-processing and automated phase reclustering so that no manual action is required in between.

Our algorithm is tested on three datasets: a cardiology dataset (ECG database) [18], an industrial network dataset for cyber attack research (SCADA dataset) [19], and a synthetic waveform dataset described in detail in Section 4.3. It turns out that, to a certain extent, our method is robust against unclean data, and the related neural networks do not show high sensitivity to the hyperparameters and are relatively easy to train.

The remainder of the paper is organised as follows. In Section 2, we specify the types of anomaly detection considered in this paper, comment on traditional methods, and introduce the concept of our solution. In Section 3, our general approach to the considered anomaly detection problems is described in detail, including data pre-processing, mathematical basis of convolutional neural networks, and training algorithm. In Section 4, our method is fine-tuned for the three aforementioned example datasets and in Section 5 the empirical results are evaluated. In Appendix A, dealing with the issue of randomly varying period length which shows up in many real-world applications such as in the ECG data (Section 4.1) and synthetic waves (Section 4.3), an auxiliary period detection scheme is designed based on classical principles of signal processing. In Appendix B, we perform some comparisons with other methods for anomaly detection in order to further highlight the advantage of using a convolutional neural network in the proposed manner.

2 Preliminaries

In preparation for the detailed description of our machine-learning phase classification scheme given in Section 3, in this section, we clarify the tasks of anomaly detection in time series with periodic characteristics, discuss some common methods, and outline the essential ideas of our approach.

2.1 Context of this work

In general, a time series $\{X_t\}_{t=0,1,2,\ldots}$ (i.e. a temporal sequence of observations $X_0, X_1, X_2, \ldots$, also termed signal) is said to exhibit periodic behaviour with *period length* s if similarities occur after every s time units, i.e. observations that are s time units apart, $X_{t_0}, X_{t_0+s}, X_{t_0+2s}, \ldots$ for any t_0, are similar.

Periodic signals occur naturally in a wide range of applications and in a large number of fields such as audio processing, vibration analysis, biomedical engineering, climatology, and economic time series analysis. Oftentimes, one wishes to monitor the behaviour of such a system. In particular, a common task when observing a signal is that of *anomaly detection*, i.e. the detection of deviations from a certain normal mode of operation. This has a variety of applications such as disease diagnosis, network security, and fraud recognition in bank transactions.

The general approach to anomaly detection is to relate a mathematical model (parametric or non-parametric) to the normal behaviour of the underlying system based on historic observations (training data) and set a confidence region for data of normal type; applying the data-adapted model to the ongoing observations (test data), whether the output lies within or outside the pre-defined confidence region decides if the corresponding input observation is considered normal or abnormal, respectively.

As with most naturally occurring signals, many of the aforementioned signals do not satisfy the exact mathematical definition of periodicity. Instead, they exhibit a property which is referred to as quasiperiodicity which basically means that the signal does not exactly repeat itself, but has deviations both in its values and in the length of the actual periods. This behaviour is very common for instance in biological or climatological systems. As a consequence for the task of anomaly detection, a sophisticated mathematical model is required to capture the essence of the diverse and noise-corrupted signals.

2.2 Tasks of anomaly detection

Mathematically, the approach to anomaly detection proposed in this paper applies to the following two types of problems:

Type A The historic observations of normal type (training data) are made up of various signals $\left\{\left\{X_t^{(\iota)}\right\}_t \mid \iota \in I \text{ (index set)}\right\}$. The signals $\left\{X_t^{(\iota)}\right\}_t$, $\iota \in I$, share certain common normal-type-characterising features, but differ in their values and exhibit periodic characteristics with individual period length $s^{(\iota)}$ which may also fluctuate over time. The task of anomaly detection in such a setting consists in rating each ongoing observation signal $\{X_t\}_t$ as normal or abnormal.

Type B The historic observations of normal type (training data) are made up of consecutive single data points $X_0, X_1, \dots, X_{N-1}$ which jointly form a time series $\{X_t\}_{t=0,\dots,N-1}$. The occurrence of the data points $X_0, \dots, X_{N-1}$ follows certain normal-type-characterising patterns, which is reflected in the corresponding time series $\{X_t\}_{t=0,\dots,N-1}$ as seasonal effects associated with period length s where s may randomly vary over time. The task of anomaly detection in such a setting consists in specifying segments of ongoing observations $\{X_t\}_{t \geq N}$ which are abnormal.

Problems of type A arise from areas such as disease diagnosis, climatology, and vibration analysis, whereas problems of type B are often addressed in the security sector and building monitoring systems within the framework of signal processing. In general, establishing an adequate mathematical model for the normal behaviour of a system requires a proper amount of training data. In our experiments in Section 4, our approach to the considered problems is applied to a cardiology dataset for detecting heart disease (cf. ECG database [18]) as an example of problems of type A, a relatively small industry dataset in the context of network security (cf. SCADA dataset [19]) as an example of problems of type B, and a more extensive synthetic waveform dataset injected with a variety of noise and anomalies (cf. Section 4.3) again as an example of problems of type B. The experimental results are provided in Section 5.

From a mathematical perspective, problems of type A are more challenging than those of type B. In the setting of type A, a considerably complex mathematical model is needed for capturing diverse variations of the normal behaviour across a variety of training signals, whereas in the setting of type B, the required mathematical model for the normal behavior is to be fitted to a single training time series. Many traditional methods for anomaly detection in periodic signals may find direct applications to problems of type B but fail to be applicable to problems of type A. This will be further discussed in the subsequent section.

2.3 On common methods

Let us comment on the adequacy of some traditional methods for detecting anomalies in periodic signals in our context.

2.3.1 Distance-analysing methods

The most straightforward treatment of seasonal data goes back to cross-correlation analysis, e.g. matrix profiles [9]. The basic idea therein is to apply a rolling window and define a Euclidean-type metric which measures the distance of consecutive values within the rolling window at different locations of the underlying time series from one another or from a fixed reference sequence (e.g. a mean window consisting of seasonal means); data points exhibiting large distance from the reference value are considered abnormal.

In general, distance-analysing approaches are not resistant against noise and fail to capture complex structures in the data. In the Appendix B, we evaluate a simple distance-based self-similarity approach in "Self-similarity approach" section. We also provide a distance-based version of our phase classification scheme (without artificial neural networks) for comparison in "Distance-based phase classification" section.

2.3.2 ARIMA methodology and Kalman filtering

A more sophisticated class of methods arises from mathematical statistics, e.g. autoregressive integrated moving average (ARIMA) methodology, methods based on structural component time series models or more general Kalman filtering (based on the linear case of the general state-space model or hidden Markov model), cf. [20, 21] for detailed description of the corresponding mathematical models. These approaches can be directly applied to problems of type B described in Section 2.2 and are based on relating a stochastic model with parameters $\Theta = \left\{\theta^1, \dots, \theta^r\right\}$ to the training part $\{X_t\}_{t=0,1,\dots,N-1}$ of the observed time series $\{X_t\}_t$ so as to make short-term (usually one-step ahead) forecasts, i.e. to estimate the conditional expectation $\mathbb{E}[X_{t+\Delta t} \mid X_t, X_{t-1}, \dots; \Theta]$ by $\hat{X}_{t+\Delta t}$ for all t (in particular for $t + \Delta t \geq N$), which basically relies on calculating the maximum likelihood estimate $\hat{\Theta}$ of the parameters Θ, making use of available observations. Setting a threshold value δ, if the actual observation X_{t_0} varies enough from the forecast value $\hat{X}_{t_0}$ in the sense that $|X_{t_0} - \hat{X}_{t_0}| > \delta$, then the data point X_{t_0} observed at time t_0 is considered abnormal.

Among the aforementioned stochastic models, the most demonstrative one is to decompose the underlying time series into trend, seasonal, and independent noise components, where the trend and seasonal components are assumed to be deterministic functions of time which can be fitted by a polynomial and conducting Fourier analysis, respectively. In fact, this is a special case of the general structural component time series model with trend and seasonal components being stochastic processes. Each structural component model can be straightforwardly represented as a linear state-space model for which Kalman filtering comes into use to generate forecasts; it also has an equivalent ARIMA model representation for which forecasting can be conducted by following the ARIMA methodology. The ARIMA approach is based on spectral theory. For seasonal time series, a parsimonious form termed (multiplicative) seasonal ARIMA (SARIMA) model may be considered. In general, modelling a time series with an ARIMA representation requires data-specific transformation (i.e. data pre-processing, e.g. logarithmising, power transformation, and differencing) and a data-adaptive hyperparameter choice (i.e. the design of the parameter set $\Theta = \{\theta^1, \ldots, \theta^r\}$, in particular the number of parameters r) which relies on inspection of the autocorrelogram and partial autocorrelogram. Each ARIMA model has an equivalent linear state-space model representation allowing Kalman filtering to be employed for forecasting.

The ARIMA approach and Kalman filtering are powerful tools in many applications and in particular in the presence of noise, provided that the hyperparameter choice is reasonable. However, being the most technically manageable segment of the general state-space models, linear models lack complexity and therefore do not always deliver a feasible approximation for real-world applications. In addition, the associated data-adapted model selection including data pre-processing requires specific expert knowledge and is therefore difficult to implement for fully automatic execution as in our machine-learning framework. Furthermore, considering problems of type A described in Section 2.2, it is unclear how to choose a general representative time series $\{X_t^*\}_t$ in which the diverse variations arising from the individual training signals $\{\{X_t^{(\iota)}\}_t \mid \iota \in I\}$ are incorporated so that the model fitted to $\{X_t^*\}_t$ applies to all normal signals.

2.3.3 *Long short-term memory units*

Long short-term memory units (LSTMs) are a special type of recurrent neural network (RNN). As such, the LSTM reads the input time series sequentially, transforming at each point in time the input data into a hidden state which is a non-linear function of the current input and the hidden state one time step earlier. The advantage of LSTMs over most other types of RNN is that the dependency of the current on the previous hidden state is designed in such a way that the LSTM obtains the ability to keep (parts of) its hidden state over a larger number of time steps than is possible with other RNN architectures, i.e. LSTMs are able to “memorise” values from the past.

Applying LSTMs to prediction tasks for the purpose of anomaly detection works in a similar manner to the application of statistical methods described in the beginning of Section 2.3.2. The main differences are that LSTMs allow for non-linear parameterisation and have the potential to support a much larger number of parameters which are not estimated directly but instead are randomly initialised at first and then optimised during training (learnt) to obtain the desired predictor. The complexity of the LSTMs allows them to ingest characteristics of rich and varied training data such as those from large training data sets of type A as described in Section 2.2 through the process of training with a stochastic gradient descent (SGD) type algorithm. The training set is processed repeatedly and the parameters of the LSTM are adjusted to optimise the quality of the forecast across the entire training dataset.

Technically, the main drawback of LSTMs is the fact that they are fundamentally still RNNs and hence also suffer from some of the difficulties typical for training this class of artificial neural network such as exploding gradients and a high potential for overfitting.

As a general drawback of using one-step ahead prediction for anomaly detection in time series, if the time series is very complex and exhibits regions in which it is difficult to make precise forecasts, such as when analysing periodic signals containing steep edges or spikes whose positions or values vary randomly over time, reliably estimating the values in these regions can actually be impossible for any type of one-step ahead prediction. It is thus difficult to derive an anomaly detector from such a predictor as the estimated values can have a large distance to the actual ones and thus show up as false positives. In Appendix B section, this is illustrated in more detail by training and evaluating an LSTM on the ECG database.

2.4 Concept of this paper

Let us now introduce the concept of our machine-learning phase classification approach to the problems specified in Section 2.2.

2.4.1 *Motivation of using convolutional neural networks for phase classification*

Convolutional neural networks (CNNs) are a specific architecture of feed-forward neural networks. When compared to a fully connected neural network, convolutional neural networks need fewer parameters. Hence, they do not require as large a training dataset and are less prone to overfitting. CNNs make explicit use of the temporal or spatial structure of the input signal; the signal is analysed locally (local receptive fields) and in a shift invariant manner (translation invariance). Investigations on the internal representations present throughout the layers of CNNs show a high tolerance to noise of various kinds.

Like LSTMs, compared to statistical methods, CNNs have the advantage that, through the use of multiple channels and non-linearities, they provide enough flexibility to capture intricate structures of analysed signals and are able to find representations for large and varied datasets. They are however easier to train than RNNs, as they suffer less from the vanishing and exploding gradient problems. The capability of a CNN of being able to process high amounts of complexity has been analysed in the field of image processing, where it was shown [22] that the neurons inside a convolutional neural network can activate on patterns ranging from simple edges to things as complex as faces.

While CNNs can be used to make forecasts in time series, they particularly excel at classification of spatial or temporal data. Since the main problem of the LSTM-based approach to anomaly detection in time series outlined above is the general unfeasability of using one-step ahead forecasts, we capitalise on the strength of CNNs in classification tasks and devise a new type of anomaly detection scheme relying on phase classification instead of one-step ahead forecasting. More details on the properties and operation of CNNs are given in Section 3.2.

2.4.2 Phase classification and anomaly detection

Motivated by the advantages of convolutional neural networks in classification tasks when dealing with spatial or temporal data, the machine-learning approach proposed in this paper is based on the following key ideas:

1. Conducting multi-dimensional time series analysis by means of multi-channel deep convolutional neural networks, where each channel in the input layer corresponds to a single feature (dimension) of the considered time series
2. Identifying phases or, equivalently, relative locations (order of occurrence) of subpatterns from time series with periodic characteristics by means of training data-adapted classifiers so that subpatterns over different periods of the underlying time series are properly separated into a certain amount of classes

To be more specific, considering a seasonal time series $\{X_t\}_t$ with period begins (e.g. time of local peak values) $\{\tau_k\}_k$, for a pre-determined *initial number of classes* n_0, sampling from the original signal n_0 overlapping segments per period with a *sliding window of length* T, each subpattern $\left\{X_t^{(m)}\right\}_{t=0,\ldots,T-1}$ with

$$X_t^{(m)} := X_{\tau_k+(\tau_{k+1}-\tau_k)(m \bmod n_0)/n_0+t}, \quad k = \lfloor m/n_0 \rfloor,$$

$m \in \mathbb{N}$, is assigned to the class labelled $m \bmod n_0$.

For seasonal data, subpatterns sampled from the time series occur repeatedly and in fixed order within each single period. A successfully trained classifier outputs the correct class indicating the phase or, equivalently, the relative location in time (i.e. time distance between subpattern and period begin) of the input subpattern. Abnormal datapoints in an input pattern are expected to cause false classification results and therefore to be identified as anomalies, which yields a direct solution to problems of type B described in Section 2.2. For problems of type A described in Section 2.2, setting a minimum expected classification accuracy (threshold value) and evaluating the classification accuracy of each test signal over a certain number of periods (which is denoted by K in the sequel), those signals that fail to achieve this minimum are considered abnormal.

In order to optimise the classification accuracy of normal data and hence prevent false-positive anomaly detection results, we carry out a dynamic reclustering which cancels confusing classes, i.e. subpatterns within a period of the signal that are similar enough to one another are merged into one class. This reclustering procedure along with the optimisation of the stride length $\Delta t := s/n_0$ (i.e. time distance between the segments to be classified) is implemented as a dynamic model selection scheme integrated in our training algorithm. In addition, we design an auxiliary period detection scheme which is employed in case of a randomly varying period length s.

The block diagram in Fig. 1 outlines the major steps of our training algorithm and anomaly detection scheme where the steps marked by dashed lines are conditioned by some model-adequacy monitoring criteria which are described in the subsequent section.

3 Method

In this section, we present the general procedure of our phase classification scheme in detail and provide some guidelines for the hyperparameter choice.

3.1 Data pre-processing

Prior to being fed into the classifier neural nets, all input signals (including training, validation, and test data) are processed by a period detector, cut into overlapping segments by a sliding window, and subsequently normalised, where the segmentation and normalisation depend on the initial number of classes n_0.

3.1.1 Period detection

In general, the seasonal effects of a time series can be recognised by examining the autocorrelogram (cf. [20, 2.1.4]) or periodogram (cf. [20, 2.2.1]). In many cases, the period length s is fixed and known. In case of a fluctuating s (cf. e.g. data from cardiology), an auxiliary period detector is designed in Appendix A, capturing the time of local extremum values (considered as period begins in our setting) $\{\tau_k\}_k$ within individual periods and using cross-correlations in order to achieve robust period detection. Note that in our setting for randomly varying period length s, the stride length $\Delta t = s/n_0$ while segmenting the signal varies proportionally to s so that the number of overlapping segments from each period is fixed and equal to n_0.

3.1.2 Sliding window

The classification accuracy of our approach turns out not to be highly sensitive to the length of the sliding window T. In the context of anomaly detection, the value of T should be kept relatively small (e.g. less than or equal to three times the average duration of a single abnormal data sequence) in order to highlight the local effect of

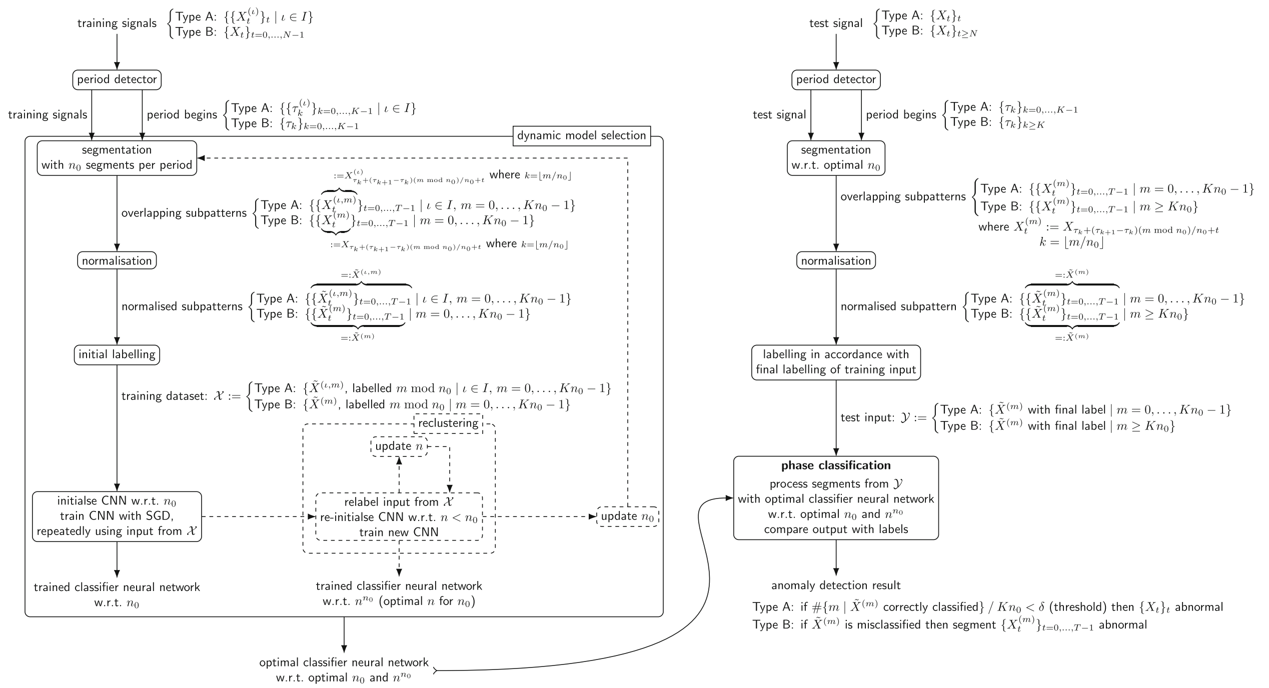
training signals $\begin{cases}\text{Type A: } \{\{X_t^{(\iota)}\}_t \mid \iota \in I\} \\ \text{Type B: } \{X_t\}_{t=0,\dots,N-1}\end{cases}$
period detector
training signals
period begins $\begin{cases}\text{Type A: } \{\{\tau_k^{(\iota)}\}_{k=0,\dots,K-1} \mid \iota \in I\} \\ \text{Type B: } \{\tau_k\}_{k=0,\dots,K-1}\end{cases}$
dynamic model selection
segmentation with n_0 segments per period
$:=X^{(\iota)}_{\tau_k+(\tau_{k+1}-\tau_k)(m \bmod n_0)/n_0+t}$ where $k=\lfloor m/n_0 \rfloor$
overlapping subpatterns $\begin{cases}\text{Type A: } \{\{X_t^{(\iota,m)}\}_{t=0,\dots,T-1} \mid \iota \in I,\, m=0,\dots,Kn_0-1\} \\ \text{Type B: } \{\{X_t^{(m)}\}_{t=0,\dots,T-1} \mid m=0,\dots,Kn_0-1\}\end{cases}$
$:=X_{\tau_k+(\tau_{k+1}-\tau_k)(m \bmod n_0)/n_0+t}$ where $k=\lfloor m/n_0 \rfloor$
normalisation
$=:\tilde{X}^{(\iota,m)}$
normalised subpatterns $\begin{cases}\text{Type A: } \{\{\tilde{X}_t^{(\iota,m)}\}_{t=0,\dots,T-1} \mid \iota \in I,\, m=0,\dots,Kn_0-1\} \\ \text{Type B: } \{\{\tilde{X}_t^{(m)}\}_{t=0,\dots,T-1} \mid m=0,\dots,Kn_0-1\}\end{cases}$
$=:\tilde{X}^{(m)}$
initial labelling
training dataset: $\mathcal{X} := \begin{cases}\text{Type A: } \{\tilde{X}^{(\iota,m)}, \text{ labelled } m \bmod n_0 \mid \iota \in I,\, m=0,\dots,Kn_0-1\} \\ \text{Type B: } \{\tilde{X}^{(m)}, \text{ labelled } m \bmod n_0 \mid m=0,\dots,Kn_0-1\}\end{cases}$
reclustering
update n
initialse CNN w.r.t. n_0 train CNN with SGD, repeatedly using input from $\mathcal{X}$
relabel input from $\mathcal{X}$ re-initialse CNN w.r.t. $n < n_0$ train new CNN
update n_0
trained classifier neural network w.r.t. n_0
trained classifier neural network w.r.t. n^{n_0} (optimal n for n_0)
optimal classifier neural network w.r.t. optimal n_0 and n^{n_0}
test signal $\begin{cases}\text{Type A: } \{X_t\}_t \\ \text{Type B: } \{X_t\}_{t\geq N}\end{cases}$
period detector
test signal
period begins $\begin{cases}\text{Type A: } \{\tau_k\}_{k=0,\dots,K-1} \\ \text{Type B: } \{\tau_k\}_{k\geq K}\end{cases}$
segmentation w.r.t. optimal n_0
overlapping subpatterns $\begin{cases}\text{Type A: } \{\{X_t^{(m)}\}_{t=0,\dots,T-1} \mid m=0,\dots,Kn_0-1\} \\ \text{Type B: } \{\{X_t^{(m)}\}_{t=0,\dots,T-1} \mid m \geq Kn_0\}\end{cases}$
where $X_t^{(m)} := X_{\tau_k+(\tau_{k+1}-\tau_k)(m \bmod n_0)/n_0+t}$ $k=\lfloor m/n_0 \rfloor$
normalisation
$=:\tilde{X}^{(m)}$
normalised subpattern $\begin{cases}\text{Type A: } \{\{\tilde{X}_t^{(m)}\}_{t=0,\dots,T-1} \mid m=0,\dots,Kn_0-1\} \\ \text{Type B: } \{\{\tilde{X}_t^{(m)}\}_{t=0,\dots,T-1} \mid m \geq Kn_0\}\end{cases}$
$=:\tilde{X}^{(m)}$
labelling in accordance with final labelling of training input
test input: $\mathcal{Y} := \begin{cases}\text{Type A: } \{\tilde{X}^{(m)} \text{ with final label} \mid m=0,\dots,Kn_0-1\} \\ \text{Type B: } \{\tilde{X}^{(m)} \text{ with final label} \mid m \geq Kn_0\}\end{cases}$
phase classification
process segments from $\mathcal{Y}$ with optimal classifier neural network w.r.t. optimal n_0 and n^{n_0} compare output with labels
anomaly detection result
Type A: if $\#\{m \mid \tilde{X}^{(m)}$ correctly classified$\}$ / $Kn_0 < \delta$ (threshold) then $\{X_t\}_t$ abnormal
Type B: if $\tilde{X}^{(m)}$ is misclassified then segment $\{X_t^{(m)}\}_{t=0,\dots,T-1}$ abnormal

Fig. 1 Block diagram of training algorithm and anomaly detection

the abnormal data points on the time series. We use a window size of $T = \lfloor 3\bar{s}/n_0 \rfloor$ (approximately three times the stride length) where $\bar{s}$ refers to the average value of s (recall that in general s may vary over time). Empirically, this has proven to be adequate for our purpose. Note that the length of the sliding window remains constant even in the case of randomly varying period length s, the varying stride length merely affects the amount of overlap between adjacent sliding windows.

3.1.3 Normalisation

In order to remove trend components and avoid skewed results due to dominating extreme values, the samples within the sliding window are normalised by adjusting the local mean and variance, that is, each time considering a d-dimensional time series $\{X_t\}_t = \left\{X_t^i\right\}_t^{i=0,\dots d-1}$ with period begins $\{\tau_k\}_k$ to be processed by a classifier neural network corresponding to initial number of classes n_0, for $i = 0, \dots, d-1$ and $m \in \mathbb{N}$, the vector $\left(\tilde{X}_t^{i,(m)}\right)_{t=0,\dots,T-1}$ is fed into channel i of the convolutional neural net, where

$$\tilde{X}_t^{i,(m)} := \frac{X_t^{i,(m)} - \mu^{i,(m)}}{\sigma^{i,(m)}} \quad \text{for} \quad t = 0, \dots, T-1$$

with

$$X_t^{i,(m)} := X^i_{\tau_k+(\tau_{k+1}-\tau_k)(m \bmod n_0)/n_0+t}, \quad k = \lfloor m/n_0 \rfloor,$$

$$\mu^{i,(m)} := \frac{1}{T}\sum_{t=0}^{T-1} X_t^{i,(m)},$$

$$\sigma^{i,(m)} := \sqrt{\frac{1}{T}\sum_{t=0}^{T-1}\left(X_t^{i,(m)} - \mu^{i,(m)}\right)^2}.$$

For the training and validation data, each subpattern $\tilde{X}^{(m)} := \{\tilde{X}_t^{(m)}\}_{t=0,\dots,T-1}$ is initially labelled $m \bmod n_0$. If reclustering occurs during the training so that the training and validation inputs are relabelled (cf. Section 3.3.3 for more details), then the test data are labelled in accordance with the final labelling of the training and validation data.

3.2 Convolutional neural networks

The core of our phase classifier is a convolutional neural network (CNN). CNNs are a special type of feedforward neural network, which exploit structures of space or time by sharing many of the weights among different neurons. We provide a short description of the mathematical basis of a convolutional neural network. For more detail on the subject, we refer the reader to the literature, e.g. [23, Ch. 9].

Basically, a feedforward neural network is a function $f\colon \mathbb{R}^{N^{(0)}} \times \mathbb{R}^P \longrightarrow \mathbb{R}^n$, mapping an input vector $x \in \mathbb{R}^{N^{(0)}}$ to an output vector $y = f(x,p) \in \mathbb{R}^n$, using a vector of parameters $p \in \mathbb{R}^P$ to adapt the mapping. When acting as a classifier, n is the number of classes and the predicted class of a given input x is taken to be $\arg\max_{j<n} y_j$. The network can be decomposed into layers, each of which represents a different function mapping vectors to vectors, i.e.

$$f(x,p) = f^{(L-1)}\left(\cdots f^{(0)}\left(x,p^{(0)}\right)\cdots, p^{(L-1)}\right)$$

where L is the number of layers and for $l = 0, \ldots, L-1$ the functions $f^{(l)} : \mathbb{R}^{N^{(l)}} \times \mathbb{R}^{P^{(l)}} \longrightarrow \mathbb{R}^{N^{(l+1)}}$ are the transformations performed by each of the single layers and the vectors $p^{(l)} \in \mathbb{R}^{P^{(l)}}$ are again parameter vectors used to adapt the mapping and given as subvectors of p. For ease of notation, let us denote the input to the function $f^{(l)}$ by $x^{(l)}$, starting with $x^{(0)} = x$ and the output of the function $f^{(l)}$ by $x^{(l+1)}$, ending with $x^{(L)} = y$.

In the most simple feedforward neural networks, each of the transformations $f^{(l)}$ is given by a multiplication with a matrix $a^{(l)} \in \mathbb{R}^{N^{(l)} \times N^{(l+1)}}$ called the *weight matrix* followed by an addition of a vector $b^{(l)} \in \mathbb{R}^{N^{(l+1)}}$ called the *bias vector* followed by the application of some non-linear function $g : \mathbb{R} \longrightarrow \mathbb{R}$ called the *activation function* to each of the components of the resulting vector, i.e.

$$\begin{aligned} x^{(l+1)} &= f^{(l)}(x^{(l)}, p^{(l)}) \\ &= g(x^{(l)} \cdot a^{(l)} + b^{(l)}) \\ &= \left(g\left(\sum_{i=0}^{N^{(l)}-1} x_i^{(l)} \cdot a_{i,j}^{(l)} + b_j^{(l)} \right) \right)_{j<N^{(l+1)}}. \end{aligned}$$

The entries of the matrix $a^{(l)}$ and the vector $b^{(l)}$ are exactly the components of the parameter vector $p^{(l)}$. This is called a *fully connected* layer.

In the case of a one-dimensional *convolutional layer*, the affine transformation $x^{(l)} \longmapsto x^{(l)} \cdot a^{(l)} + b^{(l)}$ is replaced with a more restrictive kind of affine transformation, the so-called batched convolution. For this, the vector $x^{(l)} = \left(x_i^{(l)}\right)_{i<N^{(l)}}$ is reindexed to form a two-dimensional array $\left(x_{i,t}^{(l)}\right)_{i<M^{(l)}, t<T^{(l)}}$ with $M^{(l)} \cdot T^{(l)} = N^{(l)}$, and we say that the sequence $\left(x_{i,t}^{(l)}\right)_{t<T^{(l)}}$ is fed into the ith *channel* of the convolutional layer l.[1] Similarly, the parameter vector $p^{(l)}$ is distributed not into a matrix $a^{(l)}$ and a vector $b^{(l)}$, but into a matrix of vectors $\left(k_{i,j,s}^{(l)}\right)_{i<M^{(l)}, j<M^{(l+1)}, s<S^{(l)}}$ called the *convolution kernels* and a vector $b^{(l)} \in \mathbb{R}^{M^{(l+1)}}$. The operation performed by the function $f^{(l)}$ is now given by

$$\begin{aligned} x^{(l+1)} & \\ &= f^{(l)}\left(x^{(l)}, p^{(l)}\right) \\ &= g\left(x^{(l)} \mathbin{\dot{\ast}} k^{(l)} + b^{(l)}\right) \\ &= \left(g\left(\sum_{\substack{i<M^{(l)} \\ s<S^{(l)}}} x_{i,t+S^{(l)}-1-s}^{(l)} \cdot k_{i,j,s}^{(l)} + b_j^{(l)} \right) \right)_{\substack{j<M^{(l+1)} \\ t<T^{(l+1)}}} \end{aligned}$$

and we have the constraint that $T^{(l+1)} = T^{(l)} - S^{(l)} + 1$. In many convolutional networks, the input vectors $\left(x_{i,t}^{(l)}\right)_{t<T^{(l)}}$ are extended (padded) by additional zero entries prior to being convolved. When padding with exactly $S^{(l)} - 1$ zeros, the output vectors are of the same size as the input vectors. If furthermore the padding is performed symmetrically, i.e. if $\left(S^{(l)} - 1\right)/2$ zeros are added to both ends of the signal, this is referred to as 'SAME'-padding.

We also use a type of layer called *max pooling layer* between two convolutional layers. The transfer function $f^{(l)} : \mathbb{R}^{M^{(l)} \times T^{(l)}} \longrightarrow \mathbb{R}^{M^{(l+1)} \times T^{(l+1)}}$ of this layer is given by

$$f^{(l)}\left(x^{(l)}\right) = \left(\max_{\substack{r<R^{(l)} \\ t \cdot R^{(l)} + r < T^{(l)}}} x_{j, t \cdot R^{(l)} + r}^{(l)} \right)_{\substack{j<M^{(l+1)} \\ t<T^{(l+1)}}}$$

where $R^{(l)}$ is a positive integer called the *pool size* and we have the constraints $M^{(l+1)} = M^{(l)}$ and $T^{(l+1)} = \left\lceil T^{(l)}/R^{(l)} \right\rceil$. Note that max pooling layers have no adjustable parameters $p^{(l)}$.

In our networks, we employ both convolutional and regular fully connected layers. We apply SAME-padding in all convolutional layers and use the hyperbolic tangent (tanh) as activation function g throughout the entire network, which is a common choice in feedforward neural networks.

The exact layout of the convolutional network used for our task is displayed in Table 1. Here d, T, and n denote the dimension of the input time series, the sliding window size, and the current number of classes, respectively. The layer and kernel sizes are chosen to best adapt to varying input time series dimensions, sliding window sizes, and numbers of classes. In the convolutional layers, the number of channels is increased first by a factor of 6, then by a factor of 3. Such increases are common in convolutional neural networks and allow the layers to capture different aspects of the incoming signal such as edges and more complex patterns.

Table 1 Layers of classifier neural network

	Layer Type	Sizes
0	Convolutional	$M^{(0)} = d, T^{(0)} = T,$ $S^{(0)} = (\lfloor T/6 \rfloor + 1) \cdot 2 + 1$
1	Max pooling	$M^{(1)} = M^{(0)} \cdot 6, T^{(1)} = T^{(0)},$ $R^{(1)} = 3$ if applicable, else $R^{(1)} = 1$
2	Convolutional	$M^{(2)} = M^{(1)}, T^{(2)} = \lceil T^{(1)}/R^{(1)} \rceil,$ $S^{(2)} = (\lfloor T/12 \rfloor + 1) \cdot 2 + 1$ $M^{(3)} = M^{(2)} \cdot 3, T^{(3)} = T^{(2)}$
3	Fully connected	$N^{(3)} = M^{(3)} \cdot T^{(3)}$
4	Fully connected	$N^{(4)} = \lfloor \sqrt{N^{(3)} \cdot N^{(5)}} \rfloor$
5	Output	$N^{(5)} = n$

The method, by which the parameter vector p is adjusted and the network adapts, is the minimisation of a function $h\colon \mathbb{R}^n \longrightarrow \mathbb{R}$ applied to the output of the neural network, called the *loss function*. Since we are classifying phases, to each training input x (and hence to each output y), there corresponds a label $z \in \{0, \ldots, n-1\}$. In our case, we use the cross entropy loss function, which is given by

$$h(y, z) = -w_z \log \left(\frac{\exp y_z}{\sum_{j=0}^{M-1} \exp y_j} \right)$$

where w_z denotes a weight by which the losses of each class are scaled. The weights are statically determined and are in our case chosen to be proportional to the inverse of the number of training examples for each class in order to counteract bias caused by unbalanced classes.

3.3 Training algorithm

The neural networks in our algorithm are trained by the ADAM training algorithm which is a refined version of stochastic gradient descent (SGD). In SGD, the average loss for a set $\mathcal{X}_{\text{batch}}$ containing pairs of training inputs x and corresponding labels z is minimised by changing the randomly initialised parameters p of the neural network according to the update rule

$$p \leftarrow p - \frac{\gamma}{\#\mathcal{X}_{\text{batch}}} \sum_{(x,z) \in \mathcal{X}_{\text{batch}}} \nabla_p h(f(x, p), z)$$

where γ is a tuning hyperparameter called the *learning rate* and ∇_p is the gradient operator with respect to the vector of parameters p. This minimises the average of the loss values $h(f(x, p), z)$. The gradient $\nabla_p h(f(x, p), z)$ is computed in an efficient manner via reverse-mode auto differentiation which is basically an application of the chain rule. This is also known as the backpropagation algorithm and more details on the process can be found in the literature, e.g. [23, Ch. 4]. The set $\mathcal{X}_{\text{batch}}$ is called a *mini-batch* and is taken to be a subset of the set of all available training inputs $\mathcal{X}$. The update steps are performed with changing disjoint mini-batches until the entire training dataset $\mathcal{X}$ is exhausted. Each pass through the entire set of available training data is referred to as an *epoch*. To enhance the training process (cf. [24]), for rich datasets, we change the size of the mini-batches during training, later epochs use larger mini-batch sizes. The adaptive adjustments performed by the ADAM algorithm detailed in [25] provide further enhancements to this process.

In contrast to usual classifiers, our algorithm encapsulates the gradient descent algorithm in a decision process monitoring the necessity of *dynamic reclustering* which aims to optimise the classification accuracy. The complete algorithm is given in Algorithm 1 (cf. also Fig. 1), the single steps are described in more detail in the remainder of this section.

Algorithm 1 Training algorithm

```
$n_{\text{best}} \leftarrow 0$
for $n_0 \in \{n_0^*, n_0^* - 2, \ldots, 4\}$ do
    if $n_0 \leq n_{\text{best}}$ then
        return stored net
    end if
    $n \leftarrow n_0$
    while true do
        initialise net and labels
        repeat
            perform training iteration
        until no improvement in validation loss
                                within 4 consecutive epochs
                                        ▷ training stop crit.
        if minimum class accuracy $\geq 1 - \alpha$ then
                                        ▷ reclustering stop crit.
            store net
            $n_{\text{best}} \leftarrow n$
            break
        end if
        if $n - 1 < 3$ or $n - 1 \leq n_{\text{best}}$ then
            break
        end if
        $n \leftarrow n - 1$
        recluster according to overall confusion matrix
        update weights of loss function
    end while
end for
if $n_{\text{best}} \neq 0$ then
    return stored net
else
    change $\alpha$ and rerun
end if
```

Each time having initialised the neural network for separating the currently considered classes, the gradient descent optimiser is run until a training-progress-monitoring stop criterion is fulfilled (cf. *training stop criterion* in Section 3.3.2 for more details). The classification ability of the underlying neural net is evaluated by means of the so-called *confusion matrices* (cf. Section 3.3.1) throughout the entire training. If at the end of training all classes are evaluated with sufficient accuracy (cf. *reclustering stop criterion* in Section 3.3.2), the trained neural net is stored; otherwise, a relabelling procedure according to the *overall confusion matrix* is conducted where the class with least average evaluation accuracy is merged into the class to which the corresponding inputs are most commonly misclassified during training and the neural net is re-initialised with respect to the updated classes (cf. Section 3.3.3). Among all stored neural nets, the ultimate classifier is chosen as the one having the maximum number of output classes (cf. Section 3.3.4).

In the subsequent subsections, the aforementioned reclustering process and stop criteria are described in detail.

3.3.1 Confusion matrix

In order to track the progression of classification accuracy during training, we record the *confusion matrix* evaluated on the *training data* after each epoch. For a current number of classes n and existing classes labelled as $0, \ldots, n-1$, the confusion matrix evaluated after the kth epoch is an $(n \times n)$-dimensional matrix denoted by $V_k^{(n)} := \left(V_k^{ij,(n)}\right)_{i,j=0,1,\ldots,n-1}$, where the entry $V_k^{ij,(n)}$ refers to the number of training inputs labelled as i and predicted by the neural net during the kth training epoch as class j, $k \geq 0$.

During the experimentation, we observe that classes which are easily distinguishable can already be separated after very few training iterations, whereas classes sharing more similarity perform significantly worse in the beginning and also show a slower increase of evaluation accuracy during training. Taking into account that the evaluated value of the loss function commonly follows a convex decreasing trend throughout the entire training, the above observation motivates us to assess the separation ability of the underlying neural net during training by weighting the confusion matrix with the respective contribution to the training progress and to introduce the *overall confusion matrix* denoted by $\overline{V}^{(n)} := \left(\overline{V}^{ij,(n)}\right)_{i,j=0,\ldots,n-1}$ and defined as

$$\overline{V}^{ij,(n)} := \sum_{k=1}^{E^{(n)}-1} V_k^{ij,(n)} \left(H_{k-1}^{(n)} - H_k^{(n)}\right), \tag{1}$$

where n, $E^{(n)}$, and $H_k^{(n)}$ refer to the current number of classes, the number of training epochs that are performed until the training stop criterion (cf. Section 3.3.2) is satisfied, and the average training loss during the kth epoch, respectively.

In our setting, the confusion matrices serve as the key objects of the decision criteria for our dynamic reclustering (cf. Sections 3.3.2 and 3.3.3). The definition of the overall confusion matrix in terms of (1) by taking the weighted average throughout the entire training and dropping the values from the initial epoch ($k = 0$) aims to mitigate the random effect of the initialisation of the neural network. Empirically, this yields robust reclustering results during different test runs for fixed n_0.

3.3.2 Stop criteria

The criteria for stopping the loops are related to parameterised effectiveness and accuracy requirements in the following manner:

Training stop criterion We monitor the training progress by evaluating the average loss of validation data over each training epoch. For each (re-)initialised neural network, training is stopped if no improvement in the average validation loss during the latest four epochs can be observed.

Reclustering stop criterion Allowing a *maximum per-class margin of error* $\alpha \in [0, 1)$, the reclustering procedure is stopped if

$$\min_{i=0,\ldots,n-1} \frac{V_{E^{(n)}-1}^{ii,(n)}}{\sum_{j=0}^{n-1} V_{E^{(n)}-1}^{ij,(n)}} \geq 1 - \alpha,$$

where n, $E^{(n)}$, and $V_{E^{(n)}-1}^{(n)}$ refer to the current number of classes, number of epochs for training the related network (i.e. until the training stop criterion is fulfilled), and the respective confusion matrix evaluated at the end of training (recall the definition in Section 3.3.1), respectively.

By definition of the confusion matrix in Section 3.3.1, for $k = E^{(n)} - 1$ and each $i = 0, \ldots, n-1$, the diagonal element $V_k^{ii,(n)}$ divided by the respective row sum $\sum_{j=0}^{n-1} V_k^{ij,(n)}$ of the confusion matrix is the share of correctly classified training inputs in all training inputs labelled as i evaluated during the last epoch while training the classifier neural network with n existing classes. Therefore, for a pre-defined margin of error $\alpha \in [0, 1)$, the above reclustering stop criterion requires that at the end of training the corresponding classifier neural network should correctly classify the training inputs of each existing class at least at the rate of $1 - \alpha$.

3.3.3 Reclustering

As long as the reclustering stop criterion is not fulfilled, the subsequent reclustering procedure is considered necessary.

For a current number of classes n and existing classes labelled as $0, \ldots, n-1$, let i° and j° denote the worst evaluated class, and the corresponding most misassigned class during the entire training of the respective neural net (i.e. until the training stop criterion is fulfilled) which are defined as

$$i^\circ := \arg\min_{i=0,\ldots,n-1} \frac{\overline{V}^{ii,(n)}}{\sum_{j=0}^{n-1} \overline{V}^{ij,(n)}}$$

and

$$j^\circ := \underset{\substack{j=0,\dots,n-1\\ j\neq i^\circ}}{\arg\max}\,\overline{V}^{i^\circ j,(n)}$$

respectively (recall the definition of $\overline{V}^{(n)}$ in (1)). The class labelled as i° is merged into class j°. Furthermore, since we always assume the labels to be consecutive, the training and validation inputs with the largest label $n-1$ are assigned the label of the dropped class i°.

Each time after relabelling, the weights corresponding to the remaining classes in the cost function are adjusted to be again inversely proportional to the current shares of the classes in order to warrant a well-balanced training of the updated classifier and the neural net is re-initialised.

3.3.4 Final number of classes

In the context of anomaly detection, we are dealing with the trade-off between optimising the classification accuracy of normal data preventing false positives (i.e. to cancel confusing classes) and maintaining the ability of misclassifying abnormal data for the sake of anomaly detection (i.e. to still retain sufficiently many classes characterising different phases within a period). Keeping this in mind, the final number of classes determining the ultimate classifier neural network is selected in the following manner:

Given a maximum allowed number of classes n_0^* with n_0^* an even number $n_0^* > 3$, the starting initial number of classes is set to $n_0 := n_0^*$. Each time for an updated initial number of classes n_0, the relabelling procedure described in Section 3.3.3 is run at most $(n_0 - 3)$-times (i.e. with at least 3 remaining classes). If the reclustering stop criterion is fulfilled after relabelling Δn^{n_0}-times, the *candidate final number of classes related to* n_0 is set to $n^{n_0} := n_0 - \Delta n^{n_0}$ and the corresponding neural net is stored. If $\max_{n_0'=n_0^*,\dots,n_0} n^{n_0'} \geq n_0 - 2$, the updating processes of n_0 is finished; otherwise n_0 is reduced by 2. The *overall final number of classes* refers to the maximum of n^{n_0} taken along the entire path of n_0, i.e. $\max_{n_0'=n_0^*,\dots,4} n^{n_0'}$ and the final classifier neural network is the one stored when this overall maximum was achieved. If this maximum was achieved more than once, we choose the neural network corresponding to the highest n_0 such that n^{n_0} achieved this maximum. This is because a high value of n_0 corresponds to a narrow sliding window (cf. Section 3.1.2) and hence maximises the sensitivity of the anomaly detector.

If in the end no suitable network has been stored, we increase α and rerun the algorithm.

Finally, it is worth mentioning that once all the hyperparameters are determined, the whole training algorithm introduced above, including data pre-processing and dynamic reclustering, is implemented in a machine-learning manner so that the classification and anomaly detection process can be accomplished fully automatically.

3.4 Anomaly detection

Once training is finished and in particular when the ultimate classifier neural network determined by the model selection process turns out to use initial number of classes n_0 and final number of classes n^{n_0}, each test signal is pre-processed following the procedure described in Section 3.1 with respect to n_0, labelled with respect to n^{n_0} in accordance with the training and validation data (recall the relabelling step along with the dynamic reclustering described in Section 3.3.3), and then processed by the trained ultimate classifier neural network.

For problems of type A described in Section 2.2, a minimum expected per-signal average classification accuracy δ (threshold value) should be set depending on individual needs. For instance, δ could be determined on the basis of classification accuracy on validation data. For each test signal $\{X_t\}_t$ recorded over K periods of time with period begins $\{\tau_k\}_{k=0,\dots,K-1}$, if the normalised segments $\tilde{X}^{(m)}$, $m = 0,\dots,Kn_0 - 1$ (recall Section 3.1.3), processed by the ultimate classifier neural net are evaluated with an average classification accuracy rate less than δ, i.e. if $\#\{m \mid \tilde{X}^{(m)}\text{correctly classified}\}/Kn_0 < \delta$, then the signal $\{X_t\}_t$ is considered abnormal.

Considering problems of type B described in Section 2.2, if a normalised segment $\left\{\tilde{X}_t^{(m)}\right\}_{t=0,\dots,T-1}$ (recall Section 3.1.3) of the test signal $\{X_t\}_{t\geq N}$ is misclassified by the ultimate classifier neural net, then the original segment $\left\{X_t^{(m)}\right\}_{t=0,\dots,T-1}$ with

$$X_t^{(m)} = X_{\tau_k+(\tau_{k+1}-\tau_k)(m \bmod n_0)/n_0+t}, \quad k = \lfloor m/n_0 \rfloor$$

is considered abnormal.

4 Experiments

In this section, we apply our machine-learning algorithm proposed in Section 3 to three example datasets choosing from the domains of cardiology, industry, and signal processing, aiming to show the feasibility of the method in a range of applications. The cardiology dataset is the most complex and challenging dataset representing problems of type A described in Section 2.2, as the recordings taken from healthy control patients exhibit a high level of diversity which needs to be captured by the classifier. This diversity mandates the use of a more complex representation which is one of the strengths of deep neural networks over other parametric models. The other two datasets demonstrate the applicability of the method in different contexts, including the detection of anomalies occurring only at certain instances in time and thus representing problems of type B described in Section 2.2.

4.1 Cardiology dataset

The PTB Diagnostic ECG Database is a database created by the Physikalisch-Technische Bundesanstalt (PTB) consisting of 549 electrocardiogram (ECG) recordings gathered from 290 subjects aged 18 to 87. The ECGs were recorded using a non-commercial PTB prototype recorder, the specifications of which can be found on the database website[2]. The dataset is part of PhysioNet [26] and is further described in [18].

4.1.1 Input data

We use 3/5 and 1/5 of the measurements from healthy patients for training and validation, respectively. The trained classifier is tested on the remainder of the data from healthy patients and data from all ill patients.

Due to the large data volume, we manually resample the input data to a sample rate of 50 samples per second instead of the original 1000 before feeding it into the neural network (i.e. the actual time unit applied in our training amounts to 1 time unit $=$ 20 ms). This operation is not strictly necessary, but it speeds up the training process. Also, we only use the first 60 periods of each recording during training and for testing. We train our classifier with resampled time series from healthy patients and use the data coming from all 12 conventional leads and 3 Frank leads (cf. [27]) for the ECG diagnostic, resulting in a convolutional neural net with 15 channels on the input layer.

4.1.2 Period detection

The first challenge when analysing ECG data consists in detecting the randomly varying periods of individual patients, for which we design a period detector. This detector is described in greater detail in Appendix A. The detector has a number of parameters which need to be adjusted to the dataset, the actual values used here are given in Table 2. For this dataset, the entire time series for feature ‘i’ is used as both the reference and input time series to the period detector. However, in order to ensure the requirement that no trend component exists in the signal, the first difference of the signal is used instead of the raw signal. In order to adjust for the offsets thus introduced at peak detection, between steps 4 and 5 described in Appendix A, the reference window is adjusted to be precisely centred on the corresponding peak in the original (smoothed but not differentiated) signal, i.e. its midpoint T_{k_0} is changed to

$$\underset{T_{k_0}-10\le t\le T_{k_0}+10}{\arg\max} \quad X_t.$$

The maximum allowed adjustment of 10 has empirically been found to yield satisfactory results.

The median of all observed period lengths approximately amounts to $\bar{s} = 700\,\text{ms} = 35$ time units.

Table 2 Parameters for period detector on ECG database

Parameter	Value
Prefilter window half-length n	10
Minimum base period length $s_{\min}$	500
Maximum base period length $s_{\max}$	2000
Maximum period length deviation factor σ	1/2
Reference window half-length factor λ	1/3

4.1.3 Hyperparameters

During the training, the maximum allowed number of classes and per-class margin of error are set to $n_0^* := 10$ and $\alpha := 2^{-5}$, respectively.

As per description in Table 1, each of the classifier neural networks encountered during the run consists of two convolutional layers with $M^{(0)} = 15$ and $M^{(1)} = 90$ channels, respectively, with max pooling of size $R^{(1)} = 3$ applied in between, followed by two fully connected layers, and the output layer. During the classifier selection process, the length $T^{(0)}$ of the input sequence, the kernel sizes $S^{(0)}, S^{(2)}$ of the convolutional layers, and the size $N^{(3)}$ of the first fully connected layer vary proportionally to the sliding window length $T = \lfloor 3\bar{s}/n_0 \rfloor = \lfloor 105/n_0 \rfloor$ where n_0 runs over the values in $\{10(= n_0^*), 8, 6, 4\}$ if not stopped earlier. The size $N^{(4)}$ of the second fully connected layer is determined by the geometric mean of the sizes $N^{(3)}, N^{(5)}$ of its adjacent layers and the size $N^{(5)}$ of the output layer is equal to the current number of classes n which runs over the values in $\{n_0, n_0 - 1, \ldots\}$ during the dynamic reclustering.

The ADAM optimiser with learning rate $\gamma = 0.1$ is employed for training with SGD. We start at a mini-batch size of 800 and increase it after every 2 or 3 epochs up to 4800.

4.2 SCADA dataset

In [19], Antoine Lemay and José M. Fernandez describe a simulation of an industrial control system, specifically designed for providing supervisory control and data acquisition (SCADA) network datasets for intrusion detection research. The generated datasets are openly available on GitHub[3] and contain periods of regular operation, manual interactions with the system, and anomalies caused by network intrusions. Since the operation of the simulated system is cyclic, the resulting data is mostly periodic.

4.2.1 Input data

Among the available datasets with common characteristics, we choose the first 4/5 and the last 1/5 of the dataset named ‘characterization_modbus_6RTU_with_operate’ with a duration of 5.5 min in total for training and validation, respectively, where neither the injected malicious activities nor the manual operations included are labelled, both resulting in a certain proportion of noise in the corresponding time series. The trained classifier is tested on the only three correctly labelled datasets 'moving_two_files_modbus_6RTU' ('Test Data 1'), 'CnC_uploading_exe_modbus_6RTU_with_operate' ('Test Data 2'), and 'send_a_fake_command_modbus_6RTU_with_operate' ('Test Data 3'), including no manual operations, a small portion of manual operations, and a large amount of noise, e.g. manual operations (causing non-intrusion anomalies), respectively. In each dataset, four features are considered: number and total size of sent packets, and number of active IP address and port pairs. At 1-s intervals, we record the increase in each feature and consider the corresponding 4-dimensional time series.

The given 10-s polling interval yields periodic characteristics of the considered time series with a fixed period length of $s = 10$ s.

4.2.2 Hyperparameters

We set $\alpha := 2^{-3}$ and $n_0^* := 10$ for training the classifier neural networks.

According to Table 1, all convolutional neural networks considered during the entire run include $M^{(0)} = 4$ and $M^{(1)} = 24$ channels on the first and second convolutional layers, respectively, and two fully connected layers placed between the last convolutional layer and the output layer. Considering the short input sequence length $T^{(0)} = \lfloor 3s/n_0 \rfloor = \lfloor 30/n_0 \rfloor$ with n_0 taking values in $\{10 (= n_0^*), 8, \ldots\}$, we do not apply any max pooling, i.e. $R^{(1)} = 1$. During the classifier selection process, the sizes $S^{(0)}, S^{(2)}$, and $N^{(3)}$ of the convolution kernels and the first fully connected layer, respectively, vary proportionally to the input length $T^{(0)}$. The size of the output layer $N^{(5)}$ is equal to the current number of classes n which runs over the values in $\{n_0, n_0 - 1, \ldots\}$ during the dynamic reclustering and the size of its preceding fully connected layer $N^{(4)}$ is the geometric mean of $N^{(5)}$ and $N^{(3)}$.

The ADAM optimiser with learning rate $\gamma = 0.01$ and a mini-batch size of 4 are used for training with SGD.

4.3 Wave dataset

The waves dataset is a synthetic dataset loosely modelled on a system transmitting a periodic signal. From the theory of Fourier analysis, every differentiable periodic signal $\{x_t\}_t$ with frequency f can be decomposed into its frequency components

$$x_t = a_0 + \sum_{k=1}^{\infty} a_k \cos(2\pi(fkt + \varphi_k)),$$

cf. [28, Theorem 2.1], which motivates the principal rule of our wave generator. In our consideration, the generated waves have no DC offset, i.e. $a_0 := 0$, and components only up to frequency $4f$, i.e. $a_k := 0$ for all $k \geq 5$. The signals are supposed to be transmitted over a noisy channel which we assume to add filtered Brownian and white noise. The wave generator also has some inherent randomness in the form of clock jitter, amplitude noise, and phase noise. There are also a number of fault conditions which form the basis of the anomalies to be detected.

4.3.1 Wave generator

The waves in this dataset are of the form

$$X_t = \sum_{k=1}^{4} R_t^{\mathrm{amp}k} \cos\left(2\pi\left(fkT_t + R_t^{\mathrm{ph}k}\right)\right) + R_t^{\mathrm{noise}} + N_t,$$

$t = 0, 1, 2, \ldots$, with T_t given by

$$T_t = \sum_{u=0}^{t-1} R_u^{\mathrm{time}} \quad \text{for } t = 0, 1, 2, \ldots$$

and $f := 2^{-8}$. Here, $\{N_t\}_t$ is a Gaussian white noise process, i.e. $N_0, N_1, N_2, \ldots$ are independent and identically distributed (i.i.d.) random variables with $N_t \sim \mathcal{N}\left(0, \sigma^2\right)$ for all $t = 0, 1, 2, \ldots$, and $\left\{R_t^{\mathrm{amp}k}\right\}_t$, $\left\{R_t^{\mathrm{ph}k}\right\}_t$, $\left\{R_t^{\mathrm{noise}}\right\}_t$, and $\{R_t^{\mathrm{time}}\}_t$ are independent (discrete) Ornstein-Uhlenbeck processes with individual sets of parameters. In general, an Ornstein-Uhlenbeck process $\{R_t\}_t$ obeys the stochastic differential equation

$$dR_t = \theta(\mu - R_t)\, dt + \sigma\, dW_t, \tag{2}$$

where $\mu \in \mathbb{R}$, $\sigma > 0$, $\theta \in [0, 1]$, and $\{W_t\}_t$ is a standard Brownian motion, cf. e.g. [29, Ex. 6.6]. In discrete time, a process $\{R_t\}_{t=0,1,2,\ldots}$ following (2) can be approximated by generating i.i.d. random variables $\tilde{N}_0, \tilde{N}_1, \tilde{N}_2, \ldots$ with $\tilde{N}_t \sim \mathcal{N}\left(\mu, (\sigma/\theta)^2\right)$ for all $t = 0, 1, 2, \ldots$ and exponentially smoothing them:

$$R_{t+1} := \theta\tilde{N}_t + (1 - \theta)R_t \quad \text{for } t = 0, 1, 2, \ldots. \tag{3}$$

Indeed, letting

$$N_t^* := \frac{\tilde{N}_t - \mu}{\sigma/\theta} \quad \text{for } t = 0, 1, 2, \ldots,$$

the process $\{W_t\}_{t=0,1,2,\ldots}$ with

$$W_t := \sum_{u=0}^{t-1} N_u^*$$

is a random walk with Gaussian increments and thus corresponds to a discretely sampled standard Brownian motion [30, (1.9)]. Therefore, (3) can be written as

$$\begin{aligned} R_{t+1} - R_t &= \theta\left(\mu + \frac{\sigma}{\theta} N_t^*\right) - \theta R_t \\ &= \theta(\mu - R_t) + \sigma(W_{t+1} - W_t), \end{aligned}$$

which yields a discrete counterpart of (2). The Ornstein-Uhlenbeck process can be thought of as a process performing a random walk where the increments are biased towards the mean μ. As such, it behaves locally like a Brownian motion, causing the power of the higher frequency parts of its spectrum to average $1/f^2$ (Brownian noise). The process can be used to model parameters of systems that tend to shift over time, while generally remaining close to a certain average value.

For each single wave, a set of parameters controlling the governing processes is randomly generated using the parameters in Table 3. The means of the processes for amplitude and phase variation are sampled according to the following law:

$$\log_2 \mu^{\mathrm{amp}k} \sim U(-1,1) \quad \text{and} \quad \mu^{\mathrm{ph}k} \sim U(0,1)$$

for $k = 1,2,3,4$, where $U(a,b)$ denotes the uniform distribution on the interval $[a,b)$. They remain constant throughout the wave and determine the overall shape of the wave.

4.3.2 Generated anomalies

Based on the parameters and processes employed by the wave generator, we inject the following four types of anomalies or noise:

Amplitude anomalies The amplitude process $\left\{R_t^{\mathrm{amp}k}\right\}_t$ of one of the frequency components (i.e. for a single $k \in \{1,2,3,4\}$) is increased by a, where a is randomly sampled for each anomaly according to the law $\log_2 a \sim U(1,2)$.

Phase anomalies The phase process $\left\{R_t^{\mathrm{ph}k}\right\}_t$ of one of the frequency components is changed. The amount of change is randomly sampled for each anomaly from the distribution $U(1/4, 3/4)$ resulting in a random phase change of at least 90° and at most 270°.

Pulse anomalies A pulse of random amplitude is added onto the wave. For each anomaly, the amplitude p of the pulse is randomly sampled according to the law $\log_2 p \sim U(2,4)$ and the pulse width is a random integer drawn from the interval $[2^5, 2^6)$.

White noise The white noise process $\{N_t\}_t$ is amplified by a factor α which is randomly sampled for each anomaly according to the law $\log_2 \alpha \sim U(2,6)$.

For each wave, a segment of 2^{16} samples is generated. Then 16 segments, each consisting of 2^{12} samples are generated, the last 2^{11} samples of which the anomaly or noise is injected into. For the evaluation, we use 24 generated waves, resulting in a number of 290 anomalies and 94 waves with increased white noise in the test dataset.

4.3.3 Input data and period detection

The generated waves are considered in 24 groups, where each group consists of a normal wave recorded over $2^8 = 256$ periods and further recordings, each injected with a single type of anomaly with a normal start-up time of at least $2^{11} = 2048$ time units (i.e. the first entrance time of anomalies following the respective normal wave is to the right of the time stamp $2^{11} = 2048$). In each group, we take the first $7/8$ and the remainder of the normal wave for training and validation, respectively, and subsequently test the trained classifier on the respective anomaly-injected test recordings.

Since the simulated waves contain interference in the time component which results in random period lengths s, we again make use of the period detector described in Appendix A using the parameters specified in Table 4. Note that in contrast to the treatment of ECG data, in each data group, the reference window is selected among the subpatterns extracted from the *training data*.

By construction, the average period length equals $\bar{s} = 2^8 = 256$ time units.

4.3.4 Hyperparameters

Throughout the entire training, we set the maximum number of classes and allowed per-class margin of error to $n_0^* := 10$ and $\alpha := 2^{-6}$, respectively.

As presented in Table 1, for each of the 24 waves, the corresponding classifier neural nets are all endowed with $M^{(0)} = 1$ channel and $M^{(1)} = 6$ channels on the first and second convolutional layers, respectively, where max pooling of size $R^{(1)} = 3$ is applied between the convolutional layers, and two fully connected layers are set between the last convolutional layer and the output layer. During the classifier selection process, the length $T^{(0)}$ of the input sequence, the sizes $S^{(0)}$, $S^{(2)}$ of the convolution

Table 3 Parameters for processes governing generated waves ($k = 1,2,3,4$)

Process	μ	σ	θ	R_0
$\left\{R_t^{\mathrm{time}}\right\}_t$	1	2^{-8}	2^{-8}	0
$\left\{R_t^{\mathrm{amp}k}\right\}_t$	$\mu^{\mathrm{amp}k}$	2^{-8}	2^{-8}	$\mu^{\mathrm{amp}k}$
$\left\{R_t^{\mathrm{ph}k}\right\}_t$	$\mu^{\mathrm{ph}k}$	2^{-10}	2^{-8}	$\mu^{\mathrm{ph}k}$
$\left\{R_t^{\mathrm{noise}}\right\}_t$	0	2^{-6}	2^{-8}	0
$\{N_t\}_t$	0	2^{-4}	N/A	N/A

Table 4 Parameters for period detector on wave dataset

Parameter	Value
Prefilter window half-length n	8
Minimum base period length $s_{\min}$	240
Maximum base period length $s_{\max}$	272
Maximum period length deviation factor σ	1/4
Reference window half-length factor λ	1/3

kernels, and the size $N^{(3)}$ of the first fully connected layer vary proportionally to the sliding window length $T = \lfloor 3\bar{s}/n_0 \rfloor = \lfloor 768/n_0 \rfloor$ where n_0 runs over the values in $\{10(= n_0^*), 8, 6, 4\}$ if not stopped earlier. The size $N^{(4)}$ of the second fully connected layer is the geometric mean of the sizes $N^{(3)}$ and $N^{(5)}$ of its adjacent layers and the size $N^{(5)}$ of the output layer is equal to the current number of classes n which runs over the values in $\{n_0, n_0 - 1, \ldots\}$ during the dynamic reclustering.

The ADAM optimiser with learning rate $\gamma = 0.01$ is employed for training with SGD. The mini-batch sizes are dynamically increased after every 2 or 3 epochs from 40 to 360.

5 Experimental results

In this section, we present the empirical results of the treatment of the example datasets given in Section 4 following our general phase classification scheme described in Section 3. Here, we provide both the results of selecting and training the optimal classifier neural networks and the results of anomaly detection obtained by evaluating the trained classifier neural networks on the test data (recall Section 3.4).

5.1 Cardiology dataset

The ultimate classifier resulting from the dynamic model selection process turns out to be a classifier neural network corresponding to initial number of classes $n_0 = 6$ and final number of classes $n^{n_0} = 4$, cf. Table 5 for the layout of the final CNN. The label history recorded along with the dynamic reclustering is shown in Table 6. The average validation loss recorded during the training of the respective neural nets is presented in Fig. 2. A training accuracy of 99% and a validation accuracy of 96% are achieved.

Figures 3 and 4 illustrate the result of testing the trained classifier on three patients from the category ‘healthy control’ and three ill patients: the measurements on feature ‘i’ from the test patients are presented in a temporal resolution of 20 ms and the bars in the upper and lower halves of the figures refer to the predicted classes and the true labels of the segments from the considered test signals, respectively[4].

Table 5 Layers of final classifier neural network for ECG dataset

	Layer type	Sizes
0	Convolutional	$M^{(0)} = 15, T^{(0)} = 17,$ $S^{(0)} = 7$
1	Max pooling	$M^{(1)} = 90, T^{(1)} = 17,$ $R^{(1)} = 3$
2	Convolutional	$M^{(2)} = 90, T^{(2)} = 6,$ $S^{(2)} = 5$ $M^{(3)} = 270, T^{(3)} = 6$
3	Fully connected	$N^{(3)} = 1620$
4	Fully connected	$N^{(4)} = 80$
5	Output	$N^{(5)} = 4$

Table 6 Label history

Epochs	Merge	New Labels
0 – 9	N/A	[0, 1, 2, 3, 4, 5]
9 → 10	3 to 2	[0, 1, 2, 2, 4, 3]
24 → 25	4 to 2	[0, 1, 2, 2, 2, 3]
25 – 43	N/A	[0, 1, 2, 2, 2, 3]

Figure 5 presents a statistical evaluation of the per-patient test results on patients from the seven most recorded categories in the considered database: ‘dysrhythmia’, ‘valvular heart disease’, ‘cardiomyopathy/heart failure’, ‘bundle branch block’, ‘hypertrophy’, ‘myocardial infarction’, and ‘healthy control’. The lines in different colours represent the empirical distribution functions of the per-patient classification accuracy from the aforementioned categories. Observe that the blue line related to healthy patients is located in the bottom right corner of the diagram, to the left of which all other lines corresponding to ill patients are centred (cf. the median for each category), which enables us to distinguish ill patients from healthy patients in some cases. For instance, according to the figure, if we take the average validation accuracy of 96% as the threshold for the per-patient classification accuracy, all test patients from the categories ‘dysrhythmia’ and ‘valvular heart disease’, 90% and nearly 85% of the patients from the categories ‘cardiomyopathy/heart failure’ and ‘myocardial infarction’, respectively, and over 70% of the patients from the categories ‘bundle branch block’ and ‘hypertrophy’ will be considered as anomalies, whereas up to three false-positive results (25% of) all tested patients from the category ‘healthy control’ will be assessed as normal. Since the sample sizes provided

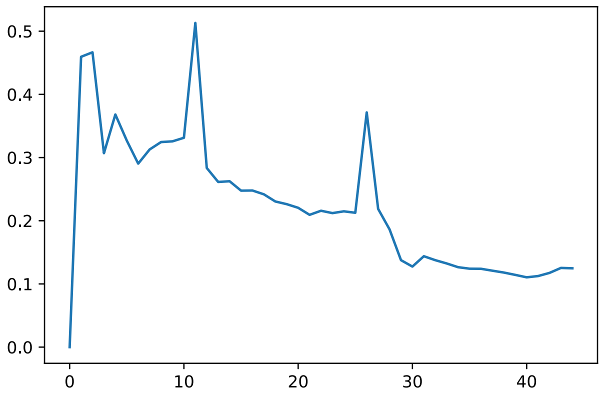

Fig. 2 Validation loss over 44 epochs for training the classifier neural nets with $n_0 = 6, n^{n_0} = 4$

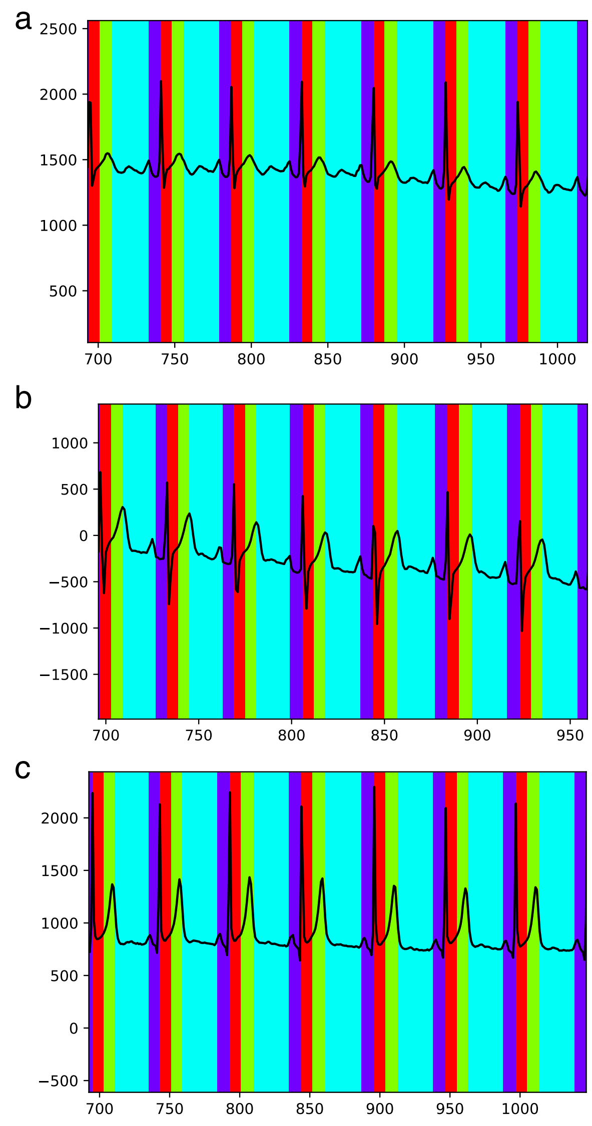

Fig. 3 Classifier applied to patients from category 'healthy control'

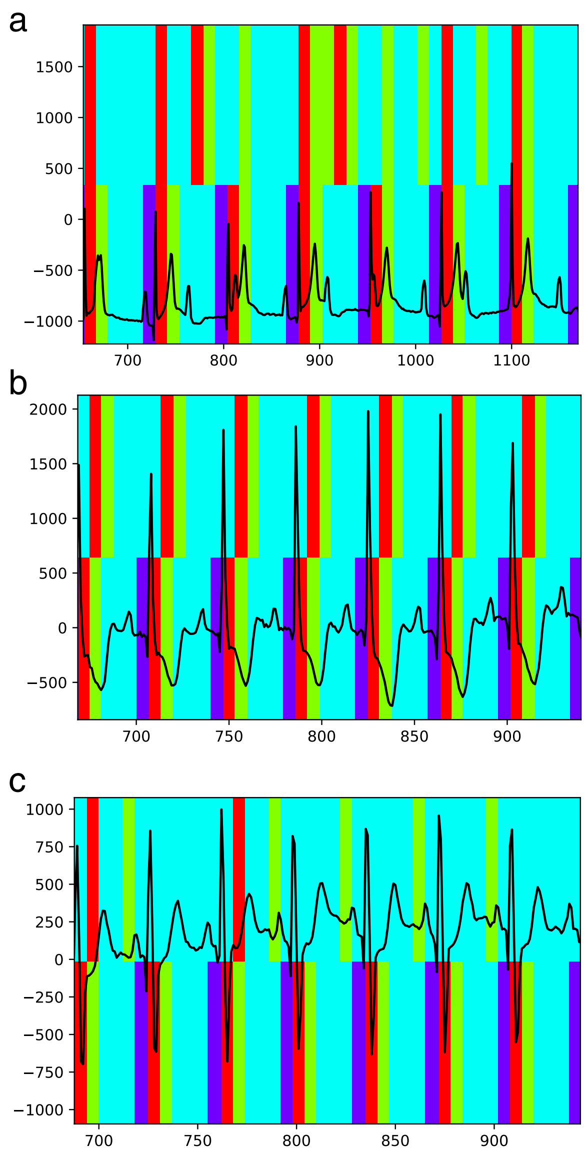

Fig. 4 Classifier applied to ill patients. **a** Classifier applied to a dysrthythmia patient, **b** Classifier applied to a valvular-heart-disease patient, **c** Classifier applied to a myocardial-infarction patient

for the individual categories vary a lot (e.g. there are 148 subjects for myocardial infarction whereas the entire category healthy control consists of only 52 subjects including training, validation and test data applied in our context), we are not in the position to make a general statement on the choice of an ideal threshold value. Table 7 provides a statistical evaluation of the per-disease average classification accuracy. It turns out that the category healthy control presents by far the best test result compared to all other categories related to heart disease (anomaly).

Note that our anomaly detection scheme does not incorporate any specific cardiological knowledge. It gives an indication whether a patient may be ill or not, it detects deviations from the known healthy data and does not classify the diseases separately. It also only gives a statistical indication, which is a result somewhat similar to the one reported in [31] where it was observed that the ECGs of ill patients showed deviations in certain affine dependencies usually present between the 12-lead and 3-lead ECGs of healthy patients.

5.2 SCADA dataset

The final classifier determined by means of the dynamic model selection scheme uses $n_0 = 10$ and $n^{n_0} = 4$, cf. Table 8 for the layout of the final CNN. The respective label history recorded during the dynamic reclustering and the evolution of the average validation loss are presented in Table 9 and Fig. 6, respectively.

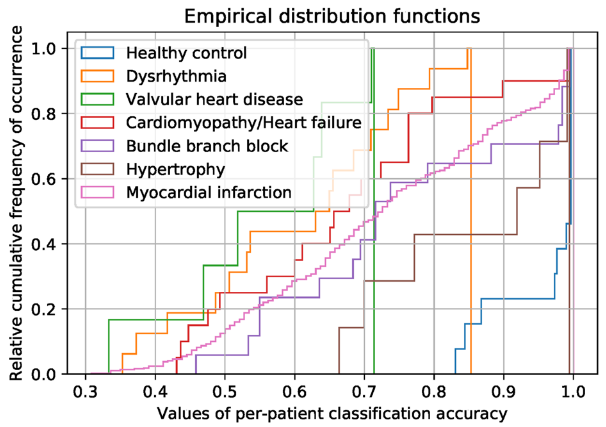

Fig. 5 Distribution of per-patient classification accuracy evaluated on test patients from different categories

Table 8 Layers of final classifier neural network for SCADA dataset

	Layer type	Sizes
0	Convolutional	$M^{(0)} = 4, T^{(0)} = 3,$ $S^{(0)} = 3$
1	Max pooling	$M^{(1)} = 24, T^{(1)} = 3,$ $R^{(1)} = 1$
2	Convolutional	$M^{(2)} = 24, T^{(2)} = 3,$ $S^{(2)} = 3$ $M^{(3)} = 72, T^{(3)} = 3$
3	Fully connected	$N^{(3)} = 216$
4	Fully connected	$N^{(4)} = 29$
5	Output	$N^{(5)} = 4$

In Fig. 7, the number of active port pairs extracted from ‘test data 1’ is plotted against time (in seconds) and the bars in the upper and lower halves represent the classes predicted by our trained neural net and the true labels of the test segments, respectively; segments which result in prediction errors are considered anomalies.

The final results of our anomaly detection algorithm on the entire test data are summarised in Table 10. In the first two (cleaner) test datasets with no or only a small amount of manual operations (noise), all cyber attacks in the test data are detected along with a single false-positive detection (corresponding to 1% false detection rate in ‘test data 1’), whereas the classifier tested on the last test dataset including a large amount of noise performs not as good, which is not surprising taking into account that only malicious activities but no manual operations or any other types of interference are labelled as anomalies and our time series analysis does not include the respective context consideration.

Indeed, the SCADA datasets which are applicable in our setting are quite small. Due to the non-compatibility between datasets with small and large amounts of noise (i.e. non-intrusion anomalies appearing in the form of pulses), it is difficult to choose one suitable dataset for training and to test the intrusion detector on datasets with incompatible characteristics, e.g. it would be unfeasible to train an anomaly detector on one of the cleaner datasets and then test it against a noisy dataset, or vice versa. For a more extensive treatment of anomaly detection of type B described in Section 2.2 using a richer dataset and the corresponding results, cf. Section 4.3 and Section 5.3.

5.3 Wave dataset

Overall, an average classification accuracy of 99% is achieved on both training and validation data.

Figures 8, 9, 10, and 11 present the detection results of our classifiers trained by individual example waves and tested on segments injected with different types of anomalies and white noise, respectively. Again, in each diagram the bars in the upper and lower halves refer to the predicted classes and true labels of the data from the test segments fed into the trained classifier, respectively. Notice that in Fig. 11, slightly increased white noise does not lead to any classification errors, which suggests some robustness property of our classifier against noise.

The final results of our anomaly detection algorithm tested on the 24 groups of synthetic waves are shown in Table 11. The amount of anomalies and white noise are obtained by counting the number of test waves injected with the respective type of interference, whereas

Table 7 Results of per-disease classification accuracy

Disease	Classification accuracy (%)
Valvular heart disease	56
Dysrhythmia	60
Cardiomyopathy/heart failure	64
Myocardial infarction	73
Bundle branch block	76
Hypertrophy	86
Healthy control	97

Table 9 Label history

Epochs	Merge	New Labels
0 – 34	N/A	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
34 → 35	1 to 4	[0, 4, 2, 3, 4, 5, 6, 7, 8, 1]
68 → 69	3 to 4	[0, 4, 2, 4, 4, 5, 6, 7, 3, 1]
92 → 93	7 to 4	[0, 4, 2, 4, 4, 5, 6, 4, 3, 1]
110 → 111	5 to 4	[0, 4, 2, 4, 4, 4, 5, 4, 3, 1]
128 → 129	2 to 4	[0, 4, 4, 4, 4, 4, 2, 4, 3, 1]
143 → 144	2 to 4	[0, 2, 2, 2, 2, 2, 2, 2, 3, 1]
144 – 160	N/A	[0, 2, 2, 2, 2, 2, 2, 2, 3, 1]

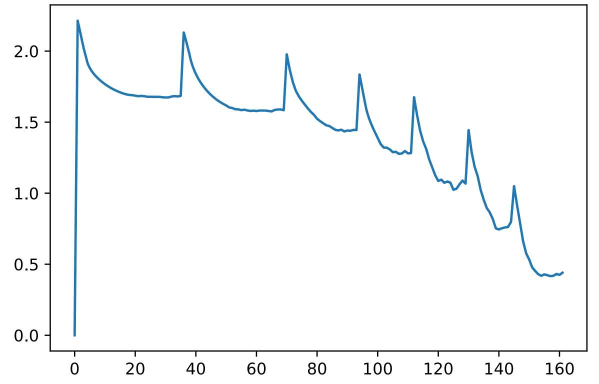

Fig. 6 Validation loss over 161 epochs for training the classifier neural nets with $n_0 = 10, n^{n_0} = 4$

the denominator for evaluating false positives equals the number of available prediction windows (test segments) in the clean test data. Overall, our algorithm yields high detection rates of all types of injected anomalies (99% on average); the small rate of false positives ($< 1\%$) confirms the model adequacy of our phase classification scheme; the low error rate in the presence of increased white noise shows the robustness of our classifier neural networks against noise to a certain extent.

6 Conclusion

In this paper, we proposed a novel approach to detecting anomalies in time series exhibiting periodic characteristics, where we applied deep convolutional neural networks for phase classification and automated phase similarity tagging. We evaluated our approach on three example datasets corresponding to the domains of cardiology, industry, and signal processing, confirming that our method is feasible in a number of contexts.

Table 10 Results of intrusion detection

Dataset	Detection rate	False positives
Test data 1	4/4	0
Test data 2	3/3	1%
Test data 3	0/1	8%

Appendix A: Period detection scheme

In this section, we provide the details for the period detection scheme used for the ECG and synthetic wave datasets. This period detection scheme is primed using a reference signal $\{Y_t^{\mathrm{raw}}\}_t$ and then applied to the actual input signal $\{X_t^{\mathrm{raw}}\}_t$. It is assumed that the input signals do not have a trend component, which can be achieved by a suitable transformation of the input signals, such as taking the first difference as in the ECG data case, cf. Section 4.1. The detection is now performed in the following steps:

1. Smooth the signals by applying a rolling mean
2. Infer approximate base period using the autocorrelation of the reference signal
3. Detect peaks in the reference signal spaced approximately one base period apart using a simple peak detection logic
4. Take the average of segments around the detected peaks and find one reference segment which most closely matches this average
5. Cross-correlate the input signal with the reference segment
6. Detect peaks in the cross-correlation spaced approximately one base period apart using again the simple peak detection logic

The steps are described in more detail in the following paragraphs.

Step 1: The raw signals $\left\{X_t^{\mathrm{raw}}\right\}_t$ and $\left\{Y_t^{\mathrm{raw}}\right\}_t$ are subjected to a rolling mean filter, resulting in smoothed signals $\{X_t\}_t$ and $\{Y_t\}_t$, respectively, i.e.

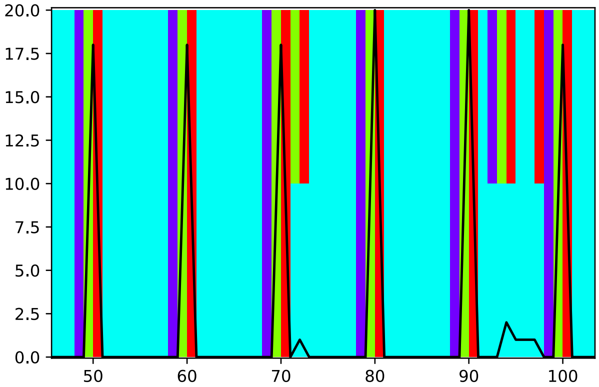

Fig. 7 Classifier applied to test data

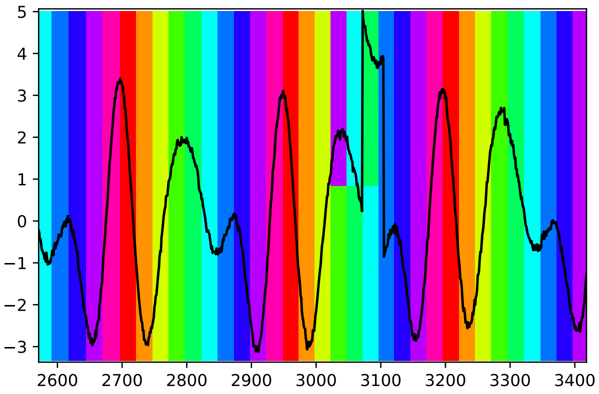

Fig. 8 Classifier with $n_0 = 10, n^{n_0} = 10$ applied to wave with pulse anomaly injected at time stamp 3072

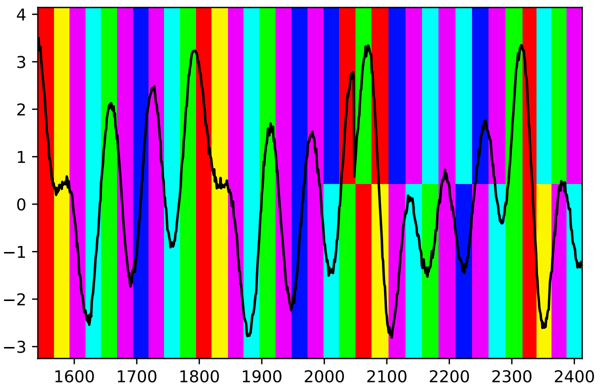

Fig. 9 Classifier with $n_0 = 10, n^{n_0} = 6$ applied to wave with abnormal phases starting at time stamp 2048

$$X_t := \frac{1}{2n+1} \sum_{k=-n}^{n} X^{\text{raw}}_{t+k}, \quad Y_t := \frac{1}{2n+1} \sum_{k=-n}^{n} Y^{\text{raw}}_{t+k}.$$

The window length $2n+1$ of this filter is chosen to provide just enough filtering to dampen some of the noise contained within the input signal.

Step 2: The sample autocorrelation $\hat{\rho}^Y_\tau$ of the (smoothed) reference signal $\{Y_t\}_t$ at lag τ is computed via

$$\hat{\rho}^Y_\tau := \frac{\hat{r}^Y_\tau}{\hat{r}^Y_0} \quad \text{for } \tau = 0, \ldots, N_Y - 1$$

with

$$\hat{r}^Y_\tau := \frac{1}{N_Y} \sum_{t=0}^{N_Y-1-\tau} (Y_{t+\tau} - \bar{Y})(Y_t - \bar{Y})$$

(cf. [20, 2.1.5]), where N_Y and $\bar{Y}$ denote the sample size and sample mean of the reference signal Y, respectively. Now the mean period length $\hat{s}$ is inferred by taking the arg max of $\hat{\rho}_\tau$ restricted to some interval $[s_{\min}, s_{\max}]$, i.e.

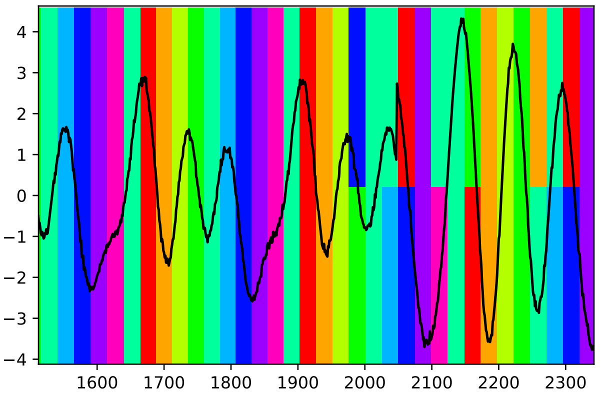

Fig. 10 Classifier with $n_0 = 10, n^{n_0} = 9$ applied to wave with abnormal amplitudes starting at time stamp 2048

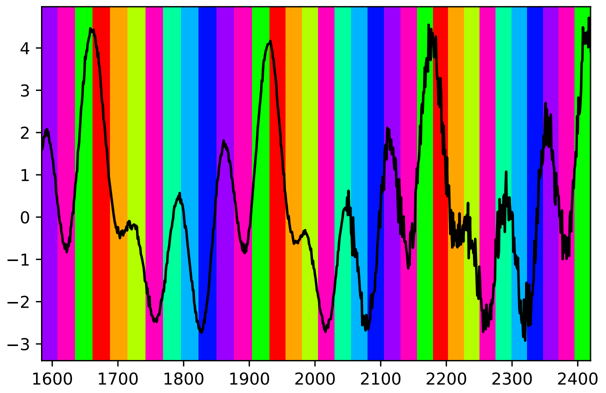

Fig. 11 Classifier with $n_0 = 10, n^{n_0} = 8$ applied to wave with slightly increased white noise ($\sigma = 4.77$) starting at time stamp 2048

$$\hat{s} := \operatorname*{arg\,max}_{s_{\min} \le \tau \le s_{\max}} \hat{\rho}^Y_\tau .$$

A plot of an example autocorrelation function is shown in Fig. 12, and the inferred mean period length is displayed by the vertical line.

Step 3: The reference signal is now fed into a simple peak detector which proceeds to inductively find peaks T_k spaced approximately one base period apart via

$$T_0 := \operatorname*{arg\,max}_{0 \le t \le \lceil \hat{s}(1+\sigma) \rceil} Y_t,$$

$$T_{k+1} := \operatorname*{arg\,max}_{T_k + \lfloor \hat{s}(1-\sigma) \rfloor \le t \le T_k + \lceil \hat{s}(1+\sigma) \rceil} Y_t,$$

where $\sigma \in [0, 1)$ is a tolerance value to account for the variability of period lengths in the signals.

Step 4: The detector now extracts subpatterns $\{U^{(k)}_t\}^{\lceil \hat{s}\lambda \rceil}_{t=\lfloor -\hat{s}\lambda \rfloor}$ from the reference signal Y_t centred at the peaks T_k, i.e. $U^{(k)}_t = Y_{T_k+t}$. Here, $\lambda \in (0, 1/2]$ is another tolerance parameter to mitigate the effects of period length variability. Then the seasonal means

$$\bar{U}_t := \frac{1}{M} \sum_{k=0}^{M-1} U^{(k)}_t$$

Table 11 Results of anomaly detection

Type	Detection Rate	%
Phases	83/85	98%
Amplitudes	102/102	100%
Pulse	102/103	99%
Total anomalies	287/290	99%
False positives	115/26798	0.43%
White noise ($\sigma \le 6$)	11/19	58%
White noise ($\sigma > 6$)	64/75	85%

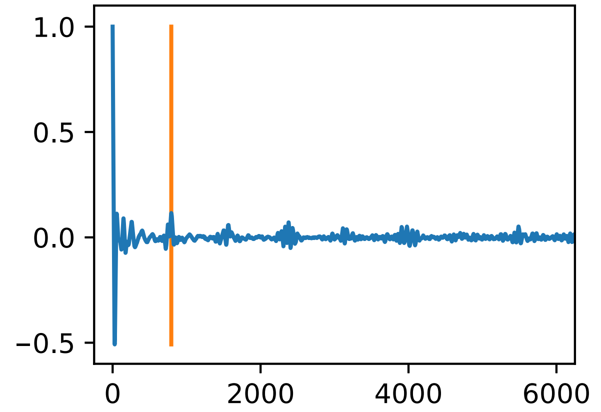

Fig. 12 Autocorrelation function of one of the ECG database records

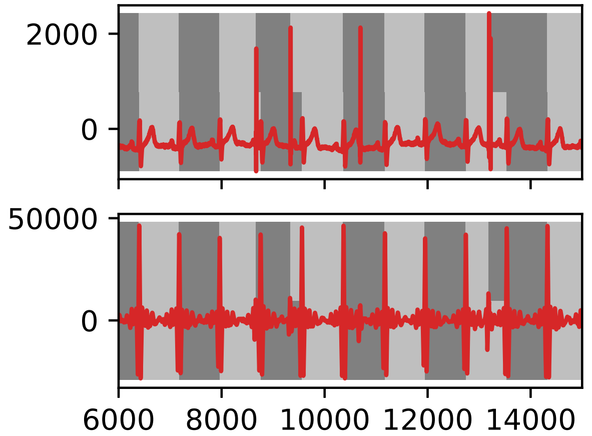

Fig. 13 Comparison of periods detected in the steps 3 and 6

are computed. Here M denotes the total number of subpatterns. Let now

$$k_0 := \arg\max_k \sum_{t=\lfloor -\hat{s}\lambda \rfloor}^{\lceil \hat{s}\lambda \rceil} U_t^{(k)} \bar{U}_t \quad \text{and} \quad U_t^{\text{ref}} := U_t^{(k_0)}.$$

The choice of k_0 ensures that $\left\{U_t^{\text{ref}}\right\}_{t=\lfloor -\hat{s}\lambda \rfloor}^{\lceil \hat{s}\lambda \rceil}$ is the subpattern with maximum similarity to the mean $\left\{\bar{U}_t\right\}_{t=\lfloor -\hat{s}\lambda \rfloor}^{\lceil \hat{s}\lambda \rceil}$ and is thus suited as a reference pattern.

Step 5: The reference pattern is now used for detecting the periods in the input signal by computing the cross-correlation function:

$$C_\tau := \left(X \star U^{\text{ref}}\right)_\tau = \sum_{t=\lfloor -\hat{s}\lambda \rfloor}^{\lceil \hat{s}\lambda \rceil} X_{\tau+t} U_t^{\text{ref}}, \quad \tau \geq 0$$

Step 6: Finally the simple peak detector from step 3 is applied to the cross-correlation $\{C_\tau\}_\tau$ to obtain the final segment beginnings.

A comparison of the periods detected by the simple peak detector from step 3 and the cross-correlating period detector from step 6 can be seen in Fig. 13. The top graph shows the input to the simple peak detector, the bottom graph shows the cross-correlation; the gray boxes in the top half of the backgrounds represent the segments inferred by the simple peak detector, those in the bottom half represent those found by the cross-correlating period detector. Notice how glitches in the input signal easily manage to confuse the simple peak detector while the cross-correlating period detector is robust to such perturbations.

Appendix B: Comparison with other methods

In this section, we perform some comparative evaluation of other methods in order to highlight in particular the utility of phase classification via convolutional neural networks for anomaly detection. We consider two classes of methods: distance-based approaches employing various types of E uclidean distance comparison (cf. "Self-similarity approach" and "Distance-based phase classification" sections) and one-step ahead forecasting (cf. "Long short-term memory predictor approach" section). In the first class of comparison, we demonstrate that even in a phase classification framework, simply comparing the Euclidean-type norm of segments of the underlying signals is less suited for capturing the essence of complex and noise corrupted data. In the second class of comparison, we show that even with the highly complex parameterisation of LSTMs, anomaly detection based on one-step ahead prediction is prone to false-positive results. Since the ECG dataset exhibits the highest level of diversity and is thus most difficult to treat among all example datasets introduced in Section 4, for this demonstration we only evaluate the reference methods on this dataset.

Self-similarity approach

One way of detecting anomalies in periodic signals is to take a sliding window of roughly one period length, normalise it, and look for a similar segment in the data preceding the window by, e.g. one to two periods. A threshold is then used to determine whether the considered window is similar enough to one of the preceding segments. This principle is used in the so-called matrix profiles approach, cf. e.g. [9]. No training data is used in this method and thus no particular characteristics of the normal data themselves are employed during the anomaly detection. The only point where training data can be useful in this approach is to determine the similarity threshold mentioned above, choosing it so as to avoid having too many false positives on non-anomalous data.

Method

Formally, if $\{X_t\}_t$ is the input signal and T is the window length, normalise each segment $X^{(\tau)} := \{X_{\tau+t}\}_{t=0,\dots,T-1}$

for $\tau \geq 0$ in an analogous manner to that of Section 3.1.3 and denote by $\tilde{X}^{(\tau)}$, $\tau \geq 0$, the respective normalised segments. Now choose a minimum shift $d_{\min}$ and a maximum shift $d_{\max}$ and compute for each $\tau \geq d_{\max}$

$$a_\tau := \min_{\tau'=\tau-d_{\max},\dots,\tau-d_{\min}} \|\tilde{X}^{(\tau')} - \tilde{X}^{(\tau)}\|^2$$

where $\|\tilde{X}^{(\tau')} - \tilde{X}^{(\tau)}\|$ denotes the Euclidean distance of $\tilde{X}^{(\tau')}$ to $\tilde{X}^{(\tau)}$, i.e.

$$\|\tilde{X}^{(\tau')} - \tilde{X}^{(\tau)}\|^2 = \sum_{t=0}^{T-1} \|\tilde{X}_t^{(\tau')} - \tilde{X}_t^{(\tau)}\|^2_{\mathbb{R}^d}.$$

a_τ is called the *self-dissimilarity* of $\{X_t\}_t$ at time τ.

Now depending on the type of problem, there are two ways to decide whether a signal is anomalous: If the task is one of type A described in Section 2.2, the average self-dissimilarity of the test signal is computed and compared against some threshold which can for instance be determined by the average self-dissimilarities of the training signals. If on the other hand the task is one of type B described Section 2.2, a threshold is chosen close to the maximum self-dissimilarity of the known normal part of the signal and the self-dissimilarity for the remaining part of the signal is compared against this threshold.

Results

For the sake of comparison, we evaluate the performance of the self-similarity-based approach on the ECG database in a similar manner as in our main result in Section 5.1 and first transform the self-dissimilarity computed as described above into a self-similarity rating via the transformation $x \mapsto 1/(x+1)$. We then average the self-similarities for each recording and plot the distributions of these averages grouped by disease. This plot is shown in Fig. 14a. One can clearly see that, apart from patients of the category 'dysrhythmia' which have the lowest self-similarity, this approach does not manage to produce any separation of ill from healthy patients.

Distance-based phase classification

A distance analysing method similar to that of "Self-similarity approach" section but more closely related to our main approach is to compute reference windows for the different phases of non-anomalous signals and use these to classify the corresponding segments of the other signals by assigning the class whose reference window has maximum similarity. Basically, this is the same method as our phase classification scheme but with the classifier neural network replaced by a simple nearest reference classifier.

Method

For this method, the same data pre-processing with respect to a chosen number of classes n_0 as described in Section 3.1 is applied to training, validation, and test signals.

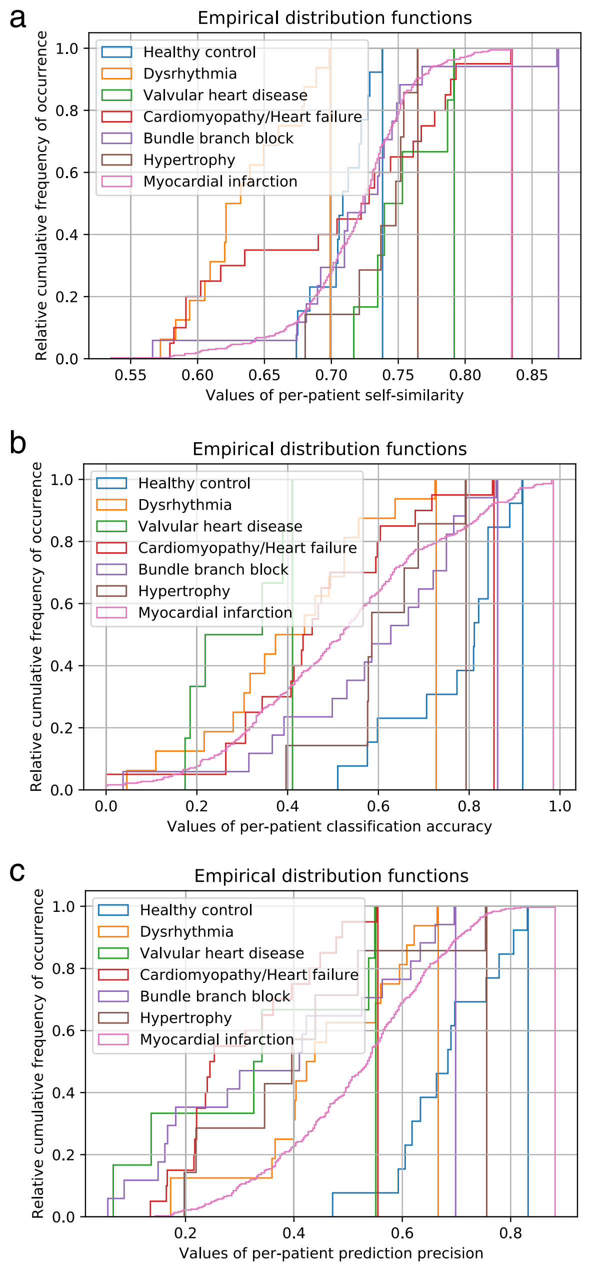

Fig. 14 Distribution of per-patient values for different comparison algorithms. **a** Distribution of per-patient self-similarity evaluated on test patients from different categories. **b** Distribution of per-patient distance-based classification accuracy evaluated on test patients from different categories. **c** Distribution of per-patient forecasting precision evaluated on test patients from different categories

For the training dataset $\mathcal{X}$ consisting of normalised segments $\tilde{X}^\theta$ labelled as belonging to class θ for $\theta = 0, \dots, n_0 - 1$ (recall the set $\mathcal{X}$ in Fig. 1 for both types A and B), we compute for each class of phase θ the seasonal averages

$$\tilde{X}^{\theta,\text{mean}} := \frac{n_0}{\#\mathcal{X}} \sum_{\tilde{X}^{\theta} \in \mathcal{X} \text{ labelled } \theta} \tilde{X}^{\theta}.$$

The classification of a normalised segment $\tilde{X}^{(m)} = \{\tilde{X}_t^{(m)}\}_{t=0,\ldots,T-1}$ (labelled m mod n_0) of a test (or validation) signal $\{X_t\}_t$ is now performed by computing

$$\arg\min_{\theta=0,\ldots,n_0-1} \|\tilde{X}^{(m)} - \tilde{X}^{\theta,\text{mean}}\|$$

where again $\|\cdot\|$ denotes the Euclidean norm as described in "Self-similarity approach" section.

The remaining part of anomaly detection is performed as in Section 3.4.

Results

To evaluate the performance of the distance-based phase classier on the ECG database, just as in our main result in Section 5.1 we record the classification accuracy on the different types of heart disease and analyse the distribution of the per-patient classification accuracy grouped by the corresponding disease. The separation into training, validation, and test data is the same as in our main experiment on the ECG database (cf. Section 4.1.1). For the number of classes, a setting of $n_0 = 6$ shows the best results, which conforms to our model selection result (cf. optimal n_0 presented in Section 5.1). The average validation accuracy amounts to 79%. The per-disease average classification accuracy is evaluated in Table 12. A plot of the distribution of per-patient classification accuracy evaluated on test patients from different categories is shown in Fig. 14b. As can be seen from both the table and the plot when compared to the results of our approach (cf. Fig. 5 and Table 7), the convolutional classifier neural network delivers generally better classification performance with far better results being obtained on the healthy control patients. In particular, we see that the blue line representing the healthy test patients in Fig. 5 is located much closer to the bottom right corner than in Fig. 14b, indicating better modelling of the normal data by our convolutional classifier neural network. An anomaly detection using the distance-based classifier thus would have a higher false positive rate than one using the convolutional neural network for classification when achieving comparable detection performance. For instance, according to Fig. 14b, if we use the average validation accuracy of 79% as the threshold value as discussed in our main result in Section 5.1, a similar detection rate in most of the ill categories but a higher false-positive rate of 38% on healthy test patients will be achieved compared to the result of our approach (25% on healthy test patients, cf. Section 5.1).

Table 12 Results of per-disease classification accuracy

Disease	Classification accuracy (%)
Valvular heart disease	28
Dysrhythmia	37
Cardiomyopathy/heart failure	44
Myocardial infarction	51
Bundle branch block	59
Hypertrophy	60
Healthy control	78

Long short-term memory predictor approach

As described in Section 2.3.3, one can use a long short-term memory unit (LSTM) to predict the signal one time step ahead, then use a threshold on the difference of this prediction to the actual signal to decide whether the signal behaves as expected or should be considered anomalous. We choose to demonstrate this method in preference to the statistical forecasting approaches mentioned in Section 2.3.2, as no further adjustment to the method is needed for handling problems of type A described in Section 2.2 and, more importantly, the forecasting performance of LSTMs on data with complex patterns has been shown to be better than that of linear models in general.

Method

Since LSTMs are a somewhat complex type of recurrent neural network, we will not describe their construction here and instead refer the reader to the literature on the subject, e.g. [32]. In our treatment of the ECG database, we use an LSTM with an input layer size of 15, a hidden layer size of 60, and an output layer size of again 15. We use a mean-squared-error loss function, discarding the first 200 predictions (4 s) to allow the LSTM to first align with the given signal. We use the same ADAM algorithm for the training of the LSTM that we also employed for training our convolutional classifier neural networks with a learning rate of $\gamma = 2^{-10}$ and 2^{10} training epochs. The separation into training, validation, and test data is also the same as in our main experiment on the ECG database (cf. Section 4.1.1).

Results

To evaluate the performance of the LSTM on the ECG database, we analyse the distribution of the forecasting precision on the patients coming from the different groups. A plot of this distribution is shown in Fig. 14c. The measure of performance used here is given by $1/(MSE+1)$ where MSE denotes the mean squared error of the predictions on the ECG recording. This transformation is applied again for the sake of easier comparability with our main result in Section 5.1. Using the same measure of performance, the prediction precision evaluated on the training and validation data are 91% and 63% on average, respectively. The large gap between the training and validation performance suggests the presence of the overfitting phenomenon mentioned in Section 2.3.3, whereas our classifier CNN approach does not suffer from this problem (see the consistency of training and validation accuracy results presented in Section 5.1). As presented in Fig. 14c, if we choose a threshold value of 63% based on the validation performance as discussed in our main result in Section 5.1, this will lead to a false positive anomaly detection rate of 31% on healthy test patients, which is higher than that of our approach (25%); at the same time, the detection performance of the LSTM-based detector is lower, with e.g. only about 75% of the patients labelled myocardial infarction (the largest category) being detected as anomalous, compared to the almost 85% of our approach. Furthermore, for illustration purposes, two examples of the predictions coming from the LSTM are displayed in Fig. 15. Notice that for both patients, the prediction fails to guess the (randomly varying) values at the spikes in the signals correctly. This (randomly) contributes to the mean squared error and thus results in a weaker separation ability of the anomaly detector.

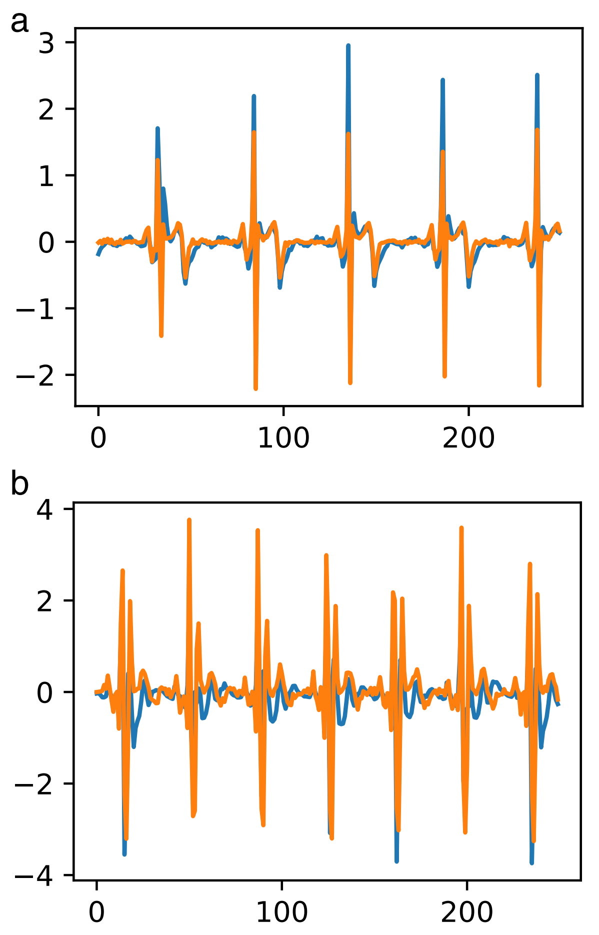

Fig. 15 Example prediction results of LSTM predictor. **a** Prediction for a healthy patient. **b** Prediction for a patient with myocardial infarction

Endnotes

[1] Vice versa, when placing a fully connected layer after a convolutional layer, the inverse reindexing is performed. When using a convolutional layer as the first layer of an artificial neural network and the input is in fact a segment of a multivariate time series $\{X_{i,t}\}_{i<M,t}$ with $M = M^{(0)}$ features, no reindexing is required. This is the case in all of our set-ups.

[2] https://physionet.org/physiobank/database/ptbdb/

[3] https://github.com/antoine-lemay/Modbus_dataset

[4] Note that here and in the sequel the coloured bars in these diagrams are always plotted between the beginnings of adjacent segments to be classified, thus only covering approximately the first third of each segment.

Abbreviations

ARIMA: Autoregressive integrated moving average; ARMA: Autoregressive–moving average; CNN: Convolutional neural network; DC: Direct current; ECG: Electrocardiogram; i.i.d.: Independent and identically distributed; IP: Internet Protocol; LSTM: Long short-term memory; MSE: Mean squared error; PTB: Physikalisch-Technische Bundesanstalt; RNN: Recurrent neural network; SARIMA: Seasonal autoregressive integrated moving average; SCADA: Supervisory control and data acquisition; SGD: Stochastic gradient descent

Acknowledgements

The authors would like to thank the anonymous referees for providing valuable suggestions which helped clarify the exposition of the material. Furthermore, the authors would like to thank Dr.-Ing. Jörg Schneider for many fruitful discussions, in particular about the cardiology dataset.

Funding

The work has been funded by the German Research Center for Artificial Intelligence and the Technical University of Kaiserslautern.

Availability of data and materials

The datasets used for the evaluation of the algorithm are available online at PhysioNet https://physionet.org/physiobank/database/ptbdb/ and the GitHub repository https://github.com/antoine-lemay/Modbus_dataset.

Authors' contributions

LA developed and implemented the core concepts of the algorithm presented within this manuscript, JA provided refinements and performed data acquisition and generation as well as further supplemental programming, HDS provided further technical knowledge and support. All authors read and approved the final manuscript.

Authors' information

LA has a Ph.D. in Mathematics, specialises in stochastic processes, stochastic filtering, and machine learning, and is currently working as a senior researcher at the German Research Center for Artificial Intelligence. JA has a Master's degree in mathematics, specialises in non-commutative harmonic analysis, has a background in digital signal processing and machine learning, and is currently working as a researcher at the German Research Center for Artificial Intelligence. HDS is the Scientific Director of the Intelligent Networks Research Group at the German Research Center for Artificial Intelligence and head of the Institute for Wireless Communication and Navigation at the Technical University of Kaiserslautern.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Author details

[1]Deutsches Forschungszentrum für Künstliche Intelligenz, Trippstadter Straße 122, 67663 Kaiserslautern, Germany. [2]Technische Universität Kaiserslautern, Paul-Ehrlich-Straße 11, 67663 Kaiserslautern, Germany.

Received: 3 December 2018 Accepted: 26 March 2019
Published online: 02 May 2019

References

1. T. T. Dang, H. Y. T. Ngan, W. Liu, in *Proceedings of the 2015 IEEE International Conference on Digital Signal Processing (DSP 2015)*. Distance-based k-nearest neighbors outlier detection method in large-scale traffic data (IEEE, Singapore, 2015), pp. 507–510
2. M. M. Breunig, H.-P. Kriegel, R. T. Ng, J. Sander, in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. LOF: Identifying Density-Based Local Outliers (SIGMOD, Dallas, 2000), pp. 93–104
3. K. Kailing, H.-P. Kriegel, P. Kröger, in *Proceedings of the 2004 SIAM International Conference on Data Mining*. Density-Connected Subspace Clustering for High-Dimensional Data (SIAM, Lake Buena Vista, 2004), pp. 246–256
4. J. Chen, S. Sathe, C. Aggarwal, D. Turaga, in *Proceedings of the 2017 SIAM International Conference on Data Mining*. Outlier Detection with Autoencoder Ensembles (SIAM, Houston, 2017), pp. 90–98
5. H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng, J. Chen, Z. Wang, H. Qiao, in *Proceedings of the 2018 World Wide Web Conference. WWW '18*. Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications (International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 2018), pp. 187–196
6. H. Louni, Outlier detection in ARMA models. J. Time Ser. Anal. **29**(6), 1057–1065 (2008)
7. J.-A. Ting, E. Theodorou, S. Schaal, in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. A Kalman filter for robust outlier detection (IEEE, San Diego, 2007), pp. 1514–1519
8. Q. Yin, L.-R. Shen, R.-B. Zhang, X.-Y. Li, H.-Q. Wang, in *Proceedings of the Second International Conference on Machine Learning and Cybernetics*. Intrusion detection based on hidden Markov model (IEEE, Xi'an, 2003), pp. 3115–3118
9. C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, E. Keogh, in *2016 IEEE 16th International Conference on Data Mining (ICDM)*. Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets (IEEE, Barcelona, 2016), pp. 1317–1322
10. F. A. Gers, J. Schmidhuber, F. Cummins, Learning to forget: continual prediction with LSTM. Neural Comput. **12**(10), 2451–2471 (2000)
11. H. Yan, H. Ouyang, Financial time series prediction based on deep learning. Wirel. Pers. Commun. **102**(2), 683–700 (2018)
12. Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Netw. **5**(2), 157–166 (1994)
13. S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, in *A Field Guide to Dynamical Recurrent Networks*, ed. by J. F. Kolen, S. C. Kremer. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies (IEEE Press, Piscataway, 2001), pp. 237–244
14. R. Pascanu, T. Mikolov, Y. Bengio, in *30th International Conference on Machine Learning, ICML*. On the difficulty of training Recurrent Neural Networks (PMLR, Atlanta, 2013), pp. 2347–2355
15. A. Krizhevsky, I. Sutskever, G. E. Hinton, in *Advances in Neural Information Processing Systems 25*, ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. ImageNet Classification with Deep Convolutional Neural Networks (Curran Associates, Inc., Lake Tahoe, 2012), pp. 1097–1105
16. C. Szegedy, S. Ioffe, V. Vanhoucke, A. A. Alemi, in *ICLR 2016 Workshop*. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning, (San Juan, 2016)
17. A. Borovykh, S. Bohte, C. W. Oosterlee, Dilated convolutional neural networks for time series forecasting. J. Comput. Finance (online early). **22**(4), 73–101 (2019)
18. R. Bousseljot, D. Kreiseler, A. Schnabel, Nutzung der EKG-Signaldatenbank CARDIODAT der PTB über das Internet. Biomed. Tech./Biomed. Eng. **40**(s1), 317–318 (1995)
19. A. Lemay, J. M. Fernandez, in *9th USENIX Workshop on Cyber Security Experimentation and Test (CSET '16)*. Providing SCADA network data sets for intrusion detection research (USENIX Association, Austin, 2016)
20. G. E. P. Box, G. M. Jenkins, G. C. Reinsel, *Time Series Analysis: Forecasting and Control. 4th ed. Wiley Series in Probability and Statistics*. (Wiley, Hoboken, 2008), p. 784
21. R. J. Elliott, L. Aggoun, J. B. Moore, *Hidden Markov Models: Estimation and Control. 1st ed. Stochastic Modelling and Applied Probability*, vol. 29. (Springer, NY, USA, 1995), p. 382
22. M. D. Zeiler, R. Fergus, in *Computer Vision – ECCV 2014, Lecture Notes in Computer Science*, ed. by D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars. Visualizing and Understanding Convolutional Networks (Springer, Cham, 2014), pp. 818–833
23. I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*. (MIT Press, Cambridge, 2016)
24. S. L. Smith, P.-J. Kindermans, Q. V. Le, in *International Conference on Learning Representations*. Don't Decay the Learning Rate, Increase the Batch Size, (Vancouver, 2018)
25. D. P. Kingma, J. L. Ba, in *International Conference on Learning Representations*. Adam: A Method for Stochastic Optimization, (San Diego, 2015)
26. A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, H. E. Stanley, PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. Circulation. **101**(23) (2000)
27. E. Frank, An Accurate, Clinically Practical System For Spatial Vectorcardiography. Circulation. **13**(5), 737–749 (1956)
28. E. M. Stein, R. Shakarchi, *Fourier Analysis: An Introduction. Princeton Lectures in Analysis, vol. 1*. (Princeton University Press, Princeton, 2003), p. 326
29. S. Shreve, *Stochastic Calculus for Finance II: Continuous-Time Models. 1st ed. Springer Finance Textbooks*. (Springer, NY, USA, 2004), p. 550
30. D. Revuz, M. Yor, *Continuous Martingales and Brownian Motion. 3rd ed. Grundlehren der mathematischen Wissenschaften*, vol. 293. (Springer, Berlin Heidelberg, 1999), p. 602
31. D. Dawson, H. Yang, M. Malshe, S. T. S. Bukkapatnam, B. Benjamin, R. Komanduri, Linear affine transformations between 3-lead (Frank XYZ leads) vectorcardiogram and 12-lead electrocardiogram signals. J. Electrocardiol. **42**(6), 622–630 (2009)
32. S. Hochreiter, J. Schmidhuber, Long Short-Term Memory. Neural Comput. **9**(8), 1735–1780 (1997)