scientific reports

Check for updates

OPEN

Machine learning-based real-time anomaly detection using data pre-processing in the telemetry of server farms

Dániel László Vajda[1]✉, Tien Van Do[1], Tamás Bérczes[2] & Károly Farkas[1]

Fast and accurate anomaly detection is critical in telemetry systems because it helps operators take appropriate actions in response to abnormal behaviours. However, recent techniques are accurate but not fast enough to deal with real-time data. There is a need to reduce the anomaly detection time, which motivates us to propose two new algorithms called AnDePeD (Anomaly Detector on Periodic Data) and AnDePed Pro. The novelty of the proposed algorithms lies in exploiting the periodic nature of data in anomaly detection. Our proposed algorithms apply a variational mode decomposition technique to find and extract periodic components from the original data before using Long Short-Term Memory neural networks to detect anomalies in the remainder time series. Furthermore, our methods include advanced techniques to eliminate prediction errors and automatically tune operational parameters. Extensive numerical results show that the proposed algorithms achieve comparable performance in terms of Precision, Recall, F-score, and MCC metrics while outperforming most of the state-of-the-art anomaly detection approaches in terms of initialisation delay and detection delay, which is favourable for practical applications.

Keywords Server farm telemetry, Real-time anomaly detection, Periodic data, Pre-processing, AnDePeD, AnDePeD Pro, VMD, Mode removal

Nowadays, telemetry systems play an essential role in most industries and the world economy as they are deployed to collect and analyse data from real-time production and service systems for establishing and maintaining profitable and affordable operation. For example, telemetry systems can be applied for server farms that host many current and future information and communication technology software instances (e.g., 5G and 6G systems) to provide customer services to various vertical industrial sectors and cities[1–5]. It is worth emphasising that telemetry systems handle vast amounts of information. Therefore, fast and accurate anomaly detection is essential for operators to take action when anomalies happen. However, recent techniques are accurate but not fast enough to deal with real-time data. This motivates us to propose two new algorithms called AnDePed (Anomaly Detector on Periodic Data) and AnDePeD Pro. Their novelty lies in exploiting the periodic nature of data inherent in the operation of server farms to reduce anomaly detection time. Thus, these algorithms are ideal for real-time usage. The contributions of this paper are as follows.

- We recognise regular and irregular patterns in datasets collected from server farms. Regular patterns in the data are present due to load from human activities performing daily routines while irregular patterns are the consequence of anomalies in server farms. Since the frequency of irregular patterns occurring is much smaller by several orders of magnitude than regular ones, removing regular patterns could facilitate the detection of anomalies.
- Therefore, we devise and apply a discrete-time Variational Mode Decomposition (VMD)[6] procedure to remove regular patterns from the measured data. The resulting remainder time series still preserves anomalies, and thus faster operation can be achieved.
- We propose a novel real-time algorithm, AnDePeD, and its improved version, AnDePeD Pro. They combine a VMD-based data pre-processing and a Long Short-Term Memory (LSTM)[7] neural network-based anomaly detection engine. According to our knowledge, we are the first to use and combine VMD and LSTM in one

[1]Department of Networked Systems and Services, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics, Magyar tudósok krt. 2, 1117 Budapest, Hungary. [2]Faculty of Informatics, University of Debrecen, Kassai út 26, 4028 Debrecen, Hungary. ✉email: dvajda@hit.bme.hu

approach for anomaly detection. Extensive numerical results show that our proposed methods exhibit comparable detection performance in terms of Precision, Recall, F-score, and MCC metrics to state-of-the-art approaches. At the same time, our proposed algorithm (AnDePeD Pro) has the smallest maximum detection delay, which is a definite advantage for practical applications.The rest of this paper is organised as follows. In Sect. "Related works", we overview papers that are close to our work. Next, in Sect. "Method", we provide a summary about the major steps and approaches we applied to establish a framework and a testbed so our algorithms and other anomaly detection procedures can be used to collect data and detect anomalies in data from real server farms. Then, we present the steps of our proposed algorithms with arguments supporting our design decisions and an overview about designing our experiments. Next, Sect. "Results" provides a numerical comparison with the state-of-the-art algorithms. Section "Discussion" includes texts about limitations and future scopes. Finally, Section "Conclusions" ends our paper.

Related works

A quest for anomaly detection algorithms and the application of anomaly detection in various areas (production, health care, image processing) has been extensively conducted in recent years. The number of literature works is enormous (see review papers[8–13]). In recent studies, Liu et al.[14] proposed a deep learning framework for diagnosing disease and anomaly detection. Tian et al.[15] developed an unsupervised anomaly detection method using generative adversarial networks and they proposed an anomaly detection approach using spatial and temporal information in multivariate time series[16]. Shi et al.[17] presented a case study of machine learning-based anomaly detection in groundwater industry, since Liu et al.[18] published a survey about deep industrial image anomaly detection. Interestingly, a comprehensive evaluation by Schmidl et al.[12] pointed out that there is no explicit winner algorithm; the context and use case should be carefully considered. Furthermore, "simple methods yield performance almost as good as more sophisticated methods"[12,19]; thus, procedures with simple auto-configuring and self-tuning capabilities are better needed. Beyond that, most existing algorithms are not fast enough to handle vast amounts of data in the server telemetry process and work in real-time. Therefore, in what follows, to achieve reproducible results, we only provide a review of works[20–26] that could be easily implemented (or with available codes) and are suitable for anomaly detection in the server farm telemetry domain. In our experiments, we compared our proposed algorithms to these methods.

Adams and MacKay[20] developed a method called Bayesian Changepoint based on the generative parameters of time series datasets. For each new data point received, the algorithm evaluates probabilities that the current value is part of a stream of variable length. When the probability that the current data point is not in any existing stream is highest (i.e., length zero), the current value is identified as a changepoint—or, in our terminology, an anomaly. However, setting parameters in the Bayesian Changepoint method, such as the maximum allowed length of a stream or the timescale parameter for estimating the prior distribution of the changepoint, is challenging and requires understanding and experimentation. The authors present three experiments on various data showcasing the wide range of domains in which their method can be used: logs while drilling a well, stock exchange data around significant events, and data on coal mine disasters. Bayesian Changepoint was applied for server farm anomaly detection in the Numenta Anomaly Benchmark (NAB)[27].

Lavin[21] proposed the Windowed Gaussian method, which computes the mean and standard deviation in a window of the latest data points and infers the probability of the new data value arriving given the sequence of previous values, based on the assumption that data follow a normal distribution. Although Windowed Gaussian can achieve high speed with low resource demands, its lightweight design means the sliding window's size and the anomaly score threshold must be manually set, requiring expert domain knowledge. Windowed Gaussian was directly developed for NAB[27], and thus for the server farm anomaly detection domain.

Wang et al.[22] presented an online detector called Relative Entropy. The authors utilised well-established practices in hypothesis testing—so-called multinomial goodness-of-fit tests—for anomaly detection. Relative Entropy determines a null-hypothesis distribution based on past values' frequencies. Suppose the current data distribution is similar enough to this null hypothesis distribution. In that case, the null hypothesis is accepted in the current data window, i.e., the data window is considered normal. However, if the null hypothesis is rejected, an anomaly is signalled. The relative entropy measure, or in other words, the Kullback-Leibler divergence, is used to compare the probability distribution in the current window to the null hypothesis distribution. Relative Entropy was tested on data from a production environment for multi-tier web applications, where the authors claim it preserves the lightweight nature of Windowed Gaussian while improving its performance through more advanced statistical models. This claim, however, is limited by the small number of datasets the authors present results for. Additionally, being a windowed approach, Relative Entropy is limited in resolution.

Burnaev and Ishimtsev[23] published an algorithm called K Nearest Neighbours Conformal Anomaly Detection (KNN CAD). It uses a contextual approach, being a conformalised density- and distance-based engine that extracts various features from the time series to determine how much the current data point differs from previous ones. This degree of difference is called the Non-Conformity Measure. After a series of data transformations, it is calculated by taking the sum of Mahalanobis distance values to the k nearest neighbours. This measure forms the anomaly score and, thus, the basis for online anomaly detection. KNN CAD also requires manually setting hyperparameter values, such as the distance threshold for an anomaly. The method was directly designed for network anomaly detection and was evaluated in NAB[28] using their approach.

Earthgecko Skyline by Stanway and Wilson[24], is based on an ensemble of sub-detectors. Although individually simple, their collective decision—either by majority, consensus, or a fixed threshold – can identify various anomalous scenarios. Sub-detectors are statistical in type, relying on metrics such as the median of deviations, average over one hour one day ago, simple and moving averages, standard deviations, least squares methods, histograms, and combinations of these. In the implementation we tested in NAB[27], the authors use seven such sub-detectors, leading to Earthgecko Skyline signalling an anomaly when at least five sub-detectors do so. The

Skyline algorithm has since become an open-source telemetry and performance monitoring framework offering integration with a large number of industry-standard tools[29].

Lee et al.[30] proposed the Greenhouse algorithm, which combines LSTM with data management techniques for anomaly detection over time series data. Although Greenhouse still requires labelled time series data for its training, it only needs normal samples for that purpose—an approach widely referred to as 'zero positive' or semi-supervised learning—and can be trained on a relatively small dataset. After training, the algorithm can be deployed and operated as follows: The LSTM model predicts the data point for the next timestep based on previous data as input. When it arrives, the actual data point is compared to the predicted one to decide whether it constitutes an anomaly. This is called the 'look-back, predict-forward' approach and all algorithms based on Greenhouse utilise the same method.

Lee et al.[31] also proposed Real-Time Proactive Anomaly Detection for Time Series (RePAD), alleviating the need for labelled training data in the original Greenhouse algorithm. RePAD can operate completely unsupervised in real-time while also being able to adapt to changing patterns in the data. Although the ability to run without anomaly labels and prior training constitute significant improvements, not to mention its flexibility, RePAD still suffers from a host of issues, resulting in relatively poor performance. The authors improved their algorithm and released the Lightweight Real-Time Ready-to-Go Anomaly Detection Approach for the Time Series (ReRe)[32] method. ReRe employs a new detector, which only operates on normal (non-anomalous) data. The new detector runs entirely separately from the original one, and ReRe only signals an anomaly if both detectors do so, aiming to reduce false positive signals. Although successful, issues remain with the detector, such as its inability to operate continuously for many timesteps.

Vajda et al.[25] developed Alternative-ReRe (Alter-Re^2), an improved version of ReRe, that reduces resource demands while significantly improving its performance by introducing a sliding window and an ageing mechanism. The latter improvements also help manage the issue of continuous real-time operation since old data points are taken into consideration less and less as new measurements arrive before being discarded completely. Alter-Re^2, however, still requires setting manual parameters while also suffering from prediction issues regarding the LSTM model. Consequently, Farkas[26] introduced Alter-Re^2+ (AREP) as an upgrade to Alter-Re^2, which includes automatic tuning of two key hyperparameters and an offset compensation component that achieves lower prediction error by strategically retraining the LSTM model when a persistent offset is detected. However, we observed that AREP's performance—like all of its predecessors'— highly depends on the type of data under analysis.

In this paper, we propose new anomaly detection procedures that exploit the inherent periodic nature of data. We show that the initialisation delay and detection delay of our proposed algorithms are smaller than most state-of-the-art anomaly detectors.

Method

Our aim is to develop new real-time anomaly detection algorithms that can be applied in a production environment. Therefore, we established a telemetry testbed with anomaly detection components based on the OpenTelemetry (OTel)[33] framework. Figure 1 plots the main components deployed in a private server farm with commercial-off-the shelf physical servers hosting various applications (web servers, AI training, collecting and processing IoT sensor data, software development tools and platforms). Sensors and monitoring tools continuously collect various information such as server and application performance measures (power

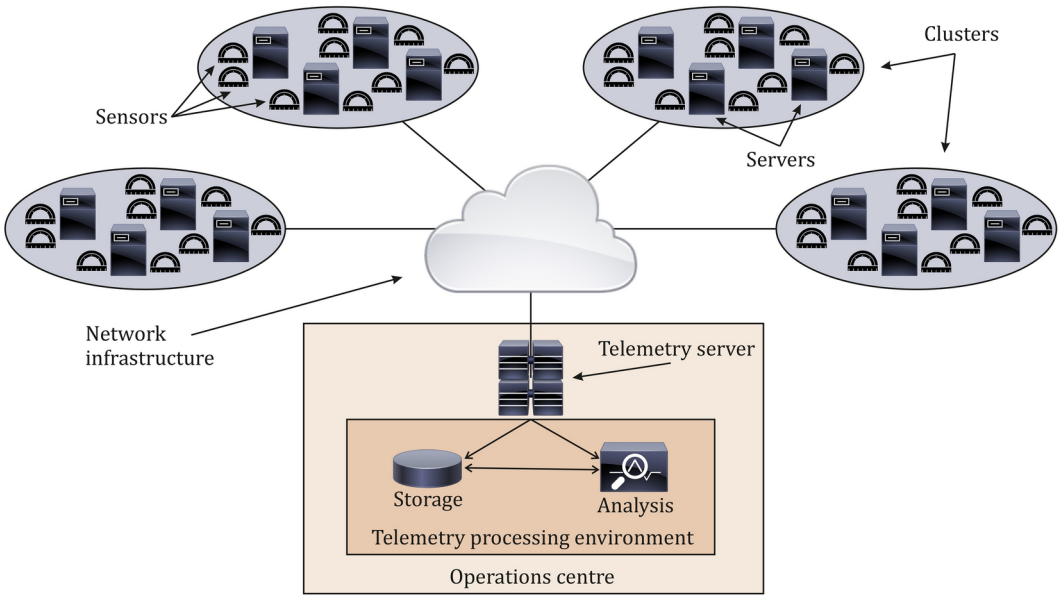


Fig. 1. General architecture of network telemetry.

consumption, CPU and memory utilisation, network traffic, disk read/write speed) and environmental conditions (temperature, humidity). Each sensor or monitoring tool normally measures quantity q, $q \in \mathcal{Q}$, at equidistant time instants $i\Delta_q$, where Δ_q is the reciprocal of the sampling frequency Ψ_q, $i \in \mathbb{Z}_{\geq 0}$ and $\mathcal{Q}$ is the set of quantities. Note that in what follows we omit q in Δ_q and use Δ to refer to the sampling time of any quantity. Measurement data is transmitted through a network to a dedicated telemetry server that is normally placed in the operations centre. The telemetry server processes, aggregates and stores the telemetry data received in a time-series database, which allows efficient querying, aggregation and data analysis in the form of time series. Note that the anomaly detection module includes a specific interface to allow the integration of any detection algorithm to the testbed. The anomaly detection module can operate in either online or offline mode. The online mode allows the detection on real-time data from server farms, while the offline mode emulates online operation using captured data. The telemetry server executes anomaly detection at time epochs $\zeta_0, \zeta_1, \ldots, \zeta_i, \zeta_{i+1}, \ldots$, $(\forall i \in \mathbb{Z}_{\geq 0})$.

We captured data from the private server farm and Amazon Web Servers, and used the offline mode to compare algorithms in terms of the traditional, well-established metrics[34] (Precision, Recall, F-score and the Matthews Correlation Coefficient). Results presented in the paper are based on data captured from Amazon Web Services (AWS), while results based on data captured in our testbed are included in the Supplementary Information as Figure S1. Since we need the knowledge (i.e., ground truth labels) of anomaly instants to compute the selected metrics, artificial anomaly points are placed in the captured data. Note that ground truth labels are not revealed to any of the anomaly detection algorithms.

Proposed algorithms—AnDePeD, AnDePeD Pro

Now we present the steps of the proposed algorithms, AnDePeD and AnDePeD Pro, that are regularly executed at time epochs $\zeta_0, \zeta_1, \ldots, \zeta_i, \zeta_{i+1}, \ldots, \forall i \in \mathbb{Z}_{\geq 0}$. These algorithms combine a VMD-based data pre-processing (see Table 1 for the notations used in VMD) with an LSTM neural network-based anomaly detection engine (see Fig. 2 for the LSTM neural network model). In the established software framework, $\zeta_{i+1} - \zeta_i$ is a constant depending on the capacity of the telemetry server and the amount of data to be processed, as an anomaly

$f(t)$	Metric or quantity associated with a server, $t \in \mathbb{R}$
Δ	Sampling time, and $\Psi = 1/\Delta$ is the sampling frequency
ζ	Latest time instant when the anomaly detector is started
L	Number of concurrently stored values
$c\Delta$	Latest measurement time instant before ζ: $c\Delta \leq \zeta < (c+1)\Delta$
$\Phi(c\Delta)$	Time series of the latest L measurement points: $\Phi(c\Delta) = \{f((c-L+1)\Delta), \ldots, f(c\Delta)\}$
K	Number of decomposed modes for VMD
$u_k(t)$	Intrinsic Mode Function (IMF), $k = 0, 1, \ldots, K-1$
$\mathbf{u}_k(c\Delta)$	Intrinsic Mode Time Series (IMTS), sampled from $u_k(t)$ and satisfying $\sum_{k=0}^{K-1} \mathbf{u}_k(c\Delta) = \Phi(c\Delta)$,
	$\mathbf{u}_k(c\Delta) = \{u_k((c-L+1)\Delta), \ldots, u_k(c\Delta)\}, k = 0, 1, \ldots, K-1$
λ	Lagrangian multipliers in time domain
$\mathscr{L}(\mathbf{u}_k(c\Delta), \Phi(c\Delta), \lambda)$	Error function to minimise that defines VMD operation
α	VMD parameter influencing the bandwidth of individual modes
∂_t	Gradient in time
j	Imaginary unit, $j^2 = -1$
$\mathscr{H}$	Operator of the discrete Hilbert transform
$\mathbf{E}$	Vector that shifts modes to baseband: $\mathbf{E} = \left\{e^{-j\omega_k(c\Delta)(c-L+1)\Delta}, \ldots, e^{-j\omega_k(c\Delta)c\Delta}\right\}$
$\omega_{k,n}(c\Delta)$	Centre frequency of the kth mode at current timestep c at VMD iteration n, $k = 0, 1, \ldots, K-1$
	($(c\Delta)$ omitted to refer to the latest available data, n omitted to refer to the final iteration)
$\Omega(c\Delta)$	Set of centre frequencies at current timestep c, $\Omega(c\Delta) = \{\omega_0(c\Delta), \omega_1(c\Delta), \ldots, \omega_{K-1}(c\Delta)\}$
$\mathbf{U}_{k,n}$	Discrete Fourier Transform (DFT) of $\mathbf{u}_k(c\Delta)$ at iteration n (n omitted to refer to the final iteration),
	$\mathbf{U}_{k,n} = \{\ldots, U_{k,n}(l/(L\Delta)), \ldots\}, l = c-L+1, \ldots, c$
$F(l/(L\Delta))$	lth point in the DFT of the measurement time series $\Phi(c\Delta)$
$\Lambda_n(l/(L\Delta))$	lth point in the DFT of Lagrangian multipliers λ
τ	Lagrangian coefficient VMD parameter (determines the speed of adjustment)
ρ	Relative magnitude of adjustment in the latest iteration
ϵ	Accuracy threshold VMD parameter
N_{iter}	Hard limit of VMD iterations

Table 1. VMD notations.

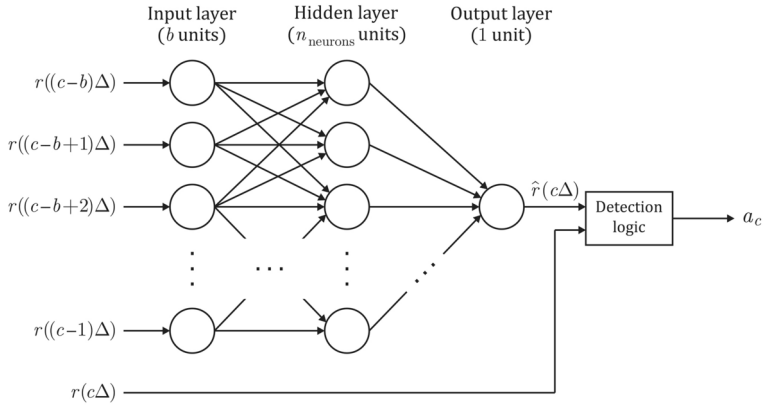


Fig. 2. Use of an LSTM neural network model for anomaly detection.

ζ_i	Time epochs when AnDePeD is executed, $i = 0, 1, \ldots$
$[\text{MIN}, \text{MAX}]$	Target range parameter for scaling
$\mathbf{s}$	Measurement time series $\Phi(c\Delta)$ scaled to range $[\text{MIN}, \text{MAX}]$, $\mathbf{s} = \{s((c-L+1)\Delta), \ldots, s(c\Delta)\}$
$\mathbf{r}$	Remainder time series, $\mathbf{r} = \mathbf{s} - \sum_k \mathbf{u}_k(c\Delta)$, $\mathbf{r} = \{r((c-L+1)\Delta), \ldots, r(c\Delta)\}$
$\widehat{\mathbf{r}}$	Predicted values of $\mathbf{r}$ by an LSTM network, $\widehat{\mathbf{r}} = \{\widehat{r}((c-L+1)\Delta), \ldots, \widehat{r}(c\Delta)\}$
$M_{\text{LSTM}}, M'_{\text{LSTM}}$	LSTM networks used for predicting time series $\widehat{\mathbf{r}}$ in AnDePeD, structure shown in Fig. 2
b	Look-back parameter, determines the number of input units in M_{LSTM} and M'_{LSTM}
	and thus the number of past data points used for prediction
n_{neurons}	Number of units used in the hidden layer of M_{LSTM} and M'_{LSTM}
n_{epochs}	Number of epochs (iterations) used during the training of M_{LSTM} and M'_{LSTM}
WS	Window Size parameter ($\text{WS} \leq L$), determines the number of values used for error calculation
AP	Age Power parameter, the larger AP is,
	the less older data points are weighted during error calculation
$\mathbf{AARE}$	Time series of Average Absolute Relative Errors calculated in AnDePeD,
	$\mathbf{AARE} = \{\text{AARE}((c-L+1)\Delta), \ldots, \text{AARE}(c\Delta)\}$
a_c	Anomaly signal, $a_c =$ `True` if an anomaly is detected at current timestep c
p_c	Pattern change signal, $p_c =$ `True` if a pattern change is detected at current timestep c
$\mu_{\text{AARE}}(c\Delta)$	Average of the previous WSAARE values at current timestep c
$\sigma_{\text{AARE}}(c\Delta)$	Standard deviation of the previous WSAARE values at current timestep c
$\text{thd}(c\Delta)$	Threshold to compare $\text{AARE}(c\Delta)$ to in order to determine a_c,
	calculated using the three-sigma rule: $\text{thd}(c\Delta) = \mu_{\text{AARE}}(c\Delta) + 3 \cdot \sigma_{\text{AARE}}(c\Delta)$
α^*, K^*	Optimal VMD parameter values achieved on the offline dataset
$\mathbf{u}^*_{k,\text{offline}}$	IMTSes calculated using α^* and K^* on the offline, stored dataset, $k = 0, 1, \ldots, K-1$
Ω_{offline}	Set of centre frequencies calculated on the offline dataset, $\Omega_{\text{offline}} = \{\omega_{0,\text{offline}}, \ldots, \omega_{K-1,\text{offline}}\}$
L_{I}	Length of the online circular buffer used in Mode-I
L_{II}	Length of the online circular buffer used in Mode-II

Table 2. AnDePeD notations.

detection procedure should be finished by a certain deadline. Table 2 summarises the notations used, while the steps of our AnDePeD method are depicted in Algorithm 2.

At each time instant ζ_i, AnDePeD applies scaling on L data points $f((c-L+1)\Delta), \ldots, f(c\Delta)$ to obtain $\mathbf{s} = \{s((c-L+1)\Delta), \ldots, s(c\Delta)\}$ so $s(i\Delta) \in [\text{MIN}, \text{MAX}]$, $\forall i = c-L+1, \ldots, c$ (the **for** loop started in line 1 of Algorithm 2). Scaling to a limited range is performed to facilitate the training of neural networks at a later stage.

Then, the proposed algorithm calls procedure VMD(s), α, K (line 4 of Algorithm 2) that takes transformed time series $\mathbf{s}$ to determine modes $\mathbf{u}_k$, $k = 0, \ldots, K-1$, where each mode is a time series with the same length L. The definitions and the foundation of the discrete-time VMD procedure (Algorithm 1) are summarised in

Tables 1 and 3. The rationale behind the application of VMD is below. It is well-known that traffic patterns from customers and users in information and communications technology systems are connected to human activities organised in work, play, and rest schedules. For example, Barlacchi et al.[35] showed that traffic produced by accessing Internet services, using mobile devices, and consuming energy and knowledge follows seasonal patterns. Therefore, server farm resource requirements (CPU, memory and storage) are inherently seasonal with daily, weekly and yearly periodicity, which is also confirmed by data collected from our private server farm and public clouds. For example, we can observe the same tendency of the curves, likely due to the periodic usage pattern, in Fig. 3 that plots CPU utilisation versus the hour of the day for six days from an Amazon Web Services (AWS)[36] server. By definition, anomalies are outlier events with irregular behaviour. Furthermore, seasonal patterns do not constitute anomalies. Therefore, our main idea is that removing periodic patterns from collected data before executing anomaly detection could reduce the detection time and the training effort the detector requires.

Given that the sum of modes only approximates the input data ($\sum_k \mathbf{u}_k \approx \mathbf{s}$), there is a slight, but important difference between this sum and the scaled original values. Remainder time series $\mathbf{r} = \{r((c-L+1)\Delta), \ldots, r(c\Delta)\}$ is equal to the scaled original data minus the sum of all modes. It is expected that all anomalies present in the scaled original data $\mathbf{s}$ will be preserved in the remainder time series $\mathbf{r}$, while most of the regular patterns are removed. The curves in Fig. 4 illustrate our procedure. The remainder time series in Fig. 4c still contains an anomaly after removing five modes (Fig. 4b) from the scaled data (Fig. 4a).

The AnDePeD algorithm predicts $\widehat{\mathbf{r}}$ time series based on remainder time series $\mathbf{r}$, (i.e., $\widehat{\mathbf{r}}$ is a future prediction of remainder time series $\mathbf{r}$). For the computation of $\widehat{\mathbf{r}}$, two LSTM networks M_{LSTM} and M'_{LSTM} with the same architecture (Fig. 2) are alternatively used depending on $\mathrm{AARE}(c\Delta)$—a time series containing the weighted errors of WS predictions, $\mu_{\mathrm{AARE}}(c\Delta)$—the average prediction error time series, and $\sigma_{\mathrm{AARE}}(c\Delta)$—the standard deviation of prediction errors. If the latest weighted prediction error is smaller than a threshold of three standard deviations, then based on the three-sigma rule of thumb it is likely that there is no anomaly, so the algorithm returns $a_c = \texttt{False}$. Otherwise, LSTM network M'_{LSTM} is trained on the previous b data points and used to recalculate the current AARE error value. If the error is still higher than the threshold, an anomaly is signalled ($a_c = \texttt{True}$). If not, data patterns have changed significantly ($p_c = \texttt{True}$), no anomaly is signalled, and the newly trained network is used for prediction (line 23 of Algorithm 2).

AnDePeD Pro combines AnDePeD and two procedures from the work of Farkas[26]. See Algorithm 3 for its steps and Table 3 for the additional notations introduced. Let

C_{corr}	Index of the timestep when the last tuning adjustment was made
m	Automatic tuning operational mode
$\mathrm{WS}_{\mathrm{low}}, \mathrm{WS}_{\mathrm{high}}$	Last correct low and high values of the window size parameter, WS
$\mathrm{AP}_{\mathrm{low}}, \mathrm{AP}_{\mathrm{high}}$	Last correct low and high values of the age power parameter, AP
$\mathbf{a}$	Boolean time series of anomaly signals, $\mathbf{a} = \{a((c-L+1)\Delta), \ldots, a(c\Delta)\}$
$\mathbf{p}$	Boolean time series of pattern change signals, $\mathbf{p} = \{p((c-L+1)\Delta), \ldots, p(c\Delta)\}$
OWS	Offset window size, $\mathrm{OWS} \leq L$
$T_{\max}$	Maximum average timestep duration parameter in seconds
OP	Percentage of allowed points above the offset
$\mathbf{o}$	Boolean time series of when offsets were registered, $\mathbf{o} = \{o((c-L+1)\Delta), \ldots, o(c\Delta)\}$
δ	Average differences between measured and predicted values time series,
	$\delta = \{\delta((c-L+1)\Delta), \ldots, \delta(c\Delta)\}$
μ_r	Average of the remainder time series ($\mathbf{r}$) in the offset window
$\mathbf{thd}_{\mathrm{off}}$	Offset thresholds time series, $\mathbf{thd}_{\mathrm{off}} = \{\mathrm{thd}_{\mathrm{off}}((c-L+1)\Delta), \ldots, \mathrm{thd}_{\mathrm{off}}(c\Delta)\}$
$\mathbf{trig}_{\mathrm{off}}$	Boolean time series recording when an LSTM retrain was triggered due to an offset
	$\mathbf{trig}_{\mathrm{off}} = \{\mathrm{trig}_{\mathrm{off}}((c-L+1)\Delta), \ldots, \mathrm{trig}_{\mathrm{off}}(c\Delta)\}$
$\bar{D}_{\mathrm{NORM}}$	Average duration of timesteps where no LSTM network retrain is triggered
$\bar{D}_{\mathrm{LSTM}}$	Average duration of LSTM retrain timesteps
start_c	Starting time of current timestep c
WS_SMALL,	Boolean values showing whether WS or AP is too small/large,
WS_LARGE,	based on the behaviour of anomaly and pattern change signals ($\mathbf{a}$ and $\mathbf{p}$)
AP_SMALL,	
AP_LARGE	
n_{AC}	Number of LSTM predictions ($\widehat{\mathbf{r}}$) above the offset threshold
n_{LSTM}	Number of timesteps in the offset window where an LSTM model retrain was triggered
D_c	Duration of the current timestep c in seconds

Table 3. AnDePeD Pro additional notations.

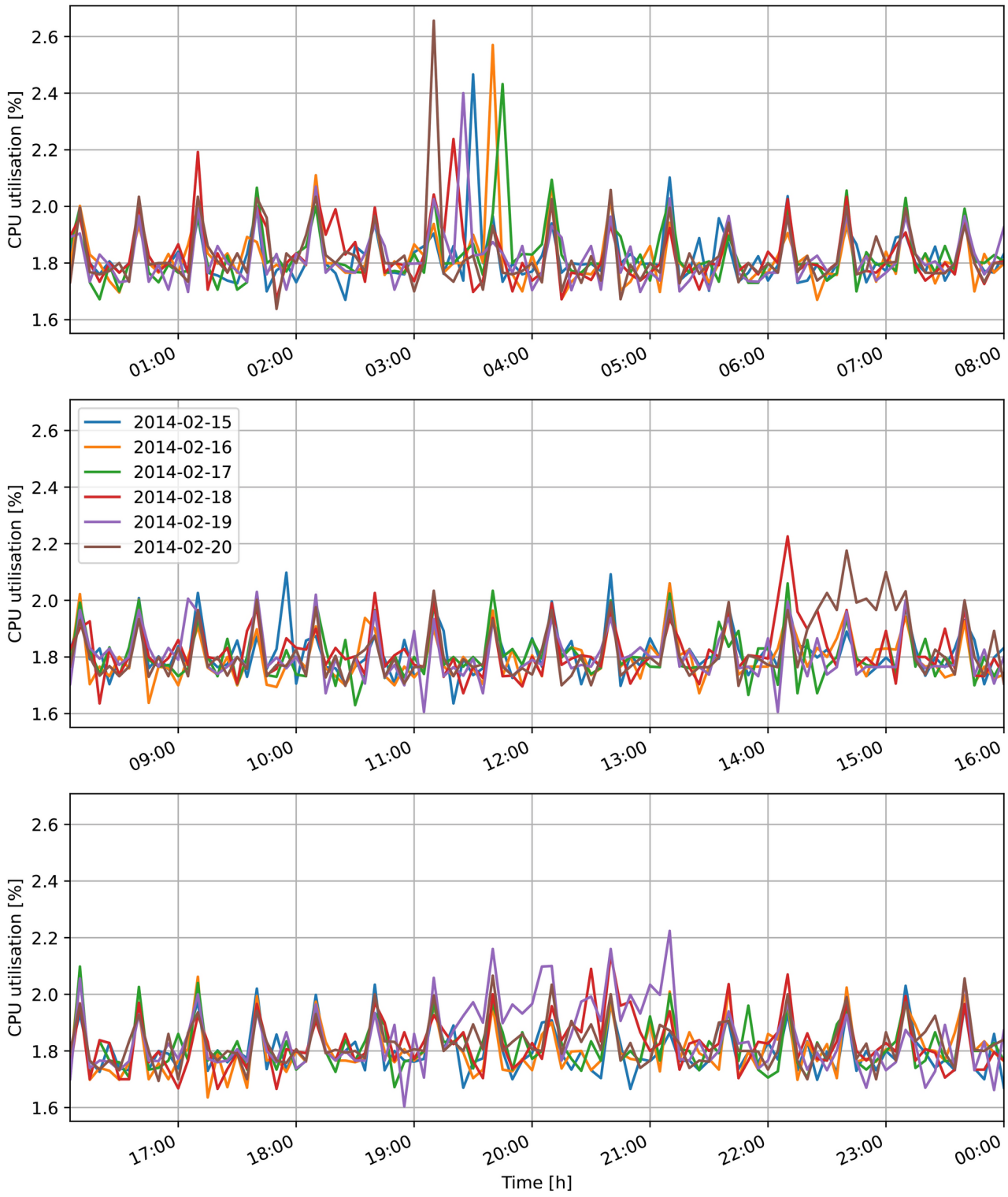
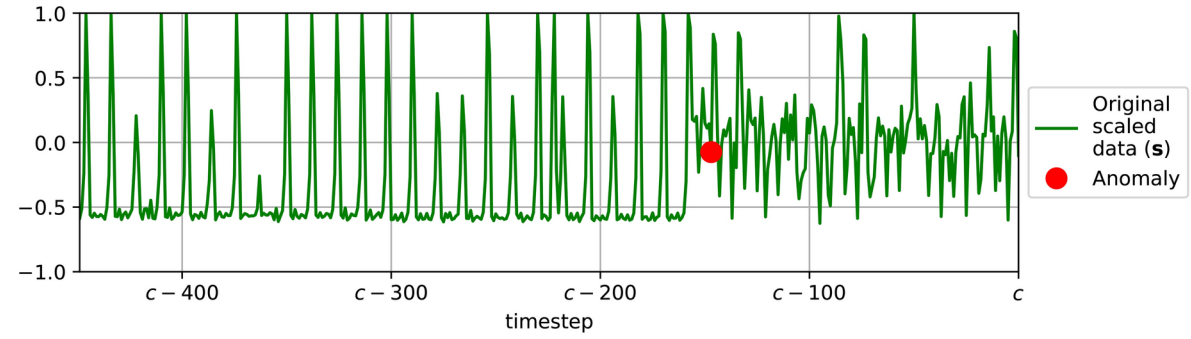


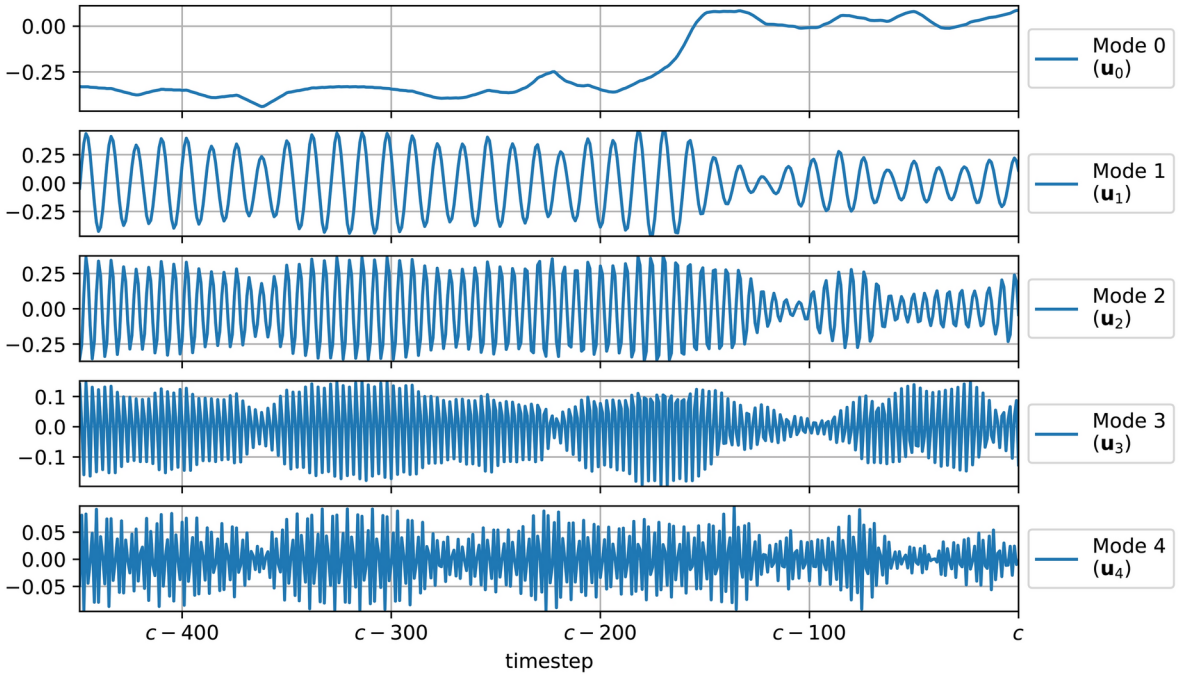
Fig. 3. Section of the Numenta Anomaly Benchmark (NAB) dataset ec2_cpu_utilization_53ea38[27], showing daily periodicity.

$\mathscr{O} = \{c, \Phi(c\Delta), \mathrm{MIN}, \mathrm{MAX}, \alpha, K, M_{\mathrm{LSTM}}, M'_{\mathrm{LSTM}}, b, \mathrm{WS}, \mathrm{AP}, \mathbf{r}, \widehat{\mathbf{r}}, \mathbf{a}, \mathbf{p}, \mathbf{AARE}, C_{\mathrm{corr}}, m, \mathrm{WS}_{\mathrm{low}}, \mathrm{WS}_{\mathrm{high}}, \mathrm{AP}_{\mathrm{low}}, \mathrm{AP}_{\mathrm{high}}, \mathrm{OWS}, T_{\mathrm{max}}, \mathrm{OP}, \mathbf{o}, \delta, \mathbf{thd}_{\mathrm{off}}, \mathbf{trig}_{\mathrm{off}}, \bar{D}_{\mathrm{NORM}}, \bar{D}_{\mathrm{LSTM}}, \zeta_c\}$ denote a subset of AnDePeD Pro algorithm parameters.

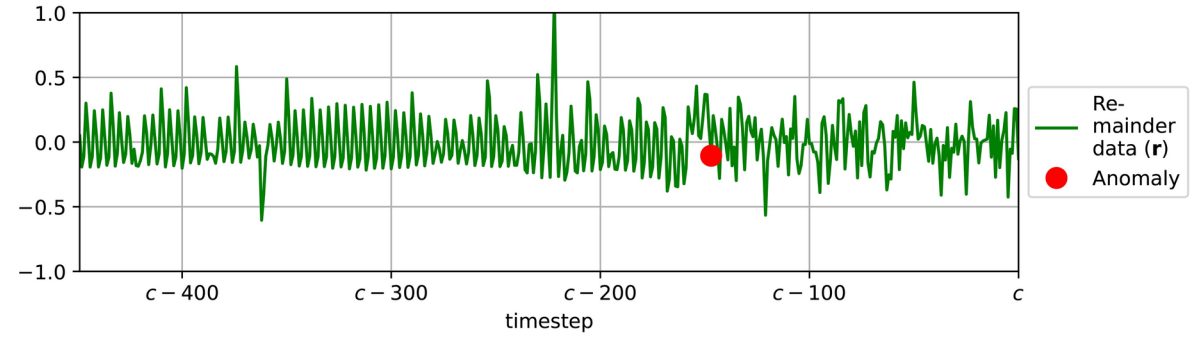
AnDePeD Pro first calls the AnDePeD procedure to perform an anomaly detection timestep as before. Selecting optimal sliding Window Size (WS) and Age Power (AP) parameters by hand or experimentally for AnDePeD is a difficult task. Each data type requires different settings, and a wrongly set Window Size or Age



(a) Original scaled dataset



(b) Decomposed modes at VMD parameters $K = 5$, $\alpha = 3700$



(c) Remainder data (original scaled data minus the sum of all modes)

Fig. 4. Example showcasing decomposition and mode removal on a section of the Numenta Anomaly Benchmark (NAB) dataset 'cpu_utilization_asg_misconfiguration'[27].

Power can significantly decrease anomaly detection performance. To avoid manual selection, AnDePeD Pro then calls the automatic tuning component of WS and AP (Supplementary Algorithm S1) based on the anomaly and pattern change signals raised by AnDePeD; a and o. More specifically, both WS and AP influence how many data points are used for calculating $\mathrm{AARE}(c\Delta)$ and $\mathrm{thd}(c\Delta)$ in AnDePeD. If too many values are used in the error, AnDePeD becomes slow to react to change, but if too few values are considered, AnDePeD becomes unstable. These phenomena can be detected by analysing the anomaly and pattern change signals and WS

and AP are raised or lowered accordingly. For the assessment of tuning criteria, Supplementary Algorithm S1 lines 4 and 18 call the method TuningCriteria( ) that returns if WS and AP are too large or small, see Tables 1 and 2 in[26] for the precise calculation.

Another issue in AnDePeD is that in some cases when a pattern change is detected, and the LSTM neural network is retrained in Algorithm 2 line 17, all subsequent LSTM predictions have a constant error compared to measured values. Let C_{train} be the timestep of retraining the LSTM network, then $|\widehat{r}(i\Delta) - r(i\Delta)| \approx \text{const.}$ $\forall i > C_{\text{train}}$. This phenomenon only occurs when data patterns change for multiple timesteps, and C_{train} occurs during this transient behaviour. Since the LSTM network is thus trained on non-representative data points, its predictions become erroneous. To solve this problem, AnDePeD Pro calls the offset compensation component (Supplementary Algorithm S2), which *i*) detects a constant offset between predicted and measured values (it is recorded in the time series o) and *ii*) triggers a new retrain for the LSTM network ($\mathbf{trig}_{\text{off}}$). Since LSTM training is the most computationally extensive task, the algorithm automatically regulates when training can occur based on the expected average duration of one timestep ($T_{\max}$). That way, the algorithm only triggers a retrain if resources allow. Note that the method NOW( ) in Algorithm 3 line 1 and Supplementary Algorithm S2 line 32 returns the current time.

Definitions

Let $f(t)$ denote a continuous time function representing a specific quantity (power consumption, CPU, memory utilisation, network traffic, etc.) associated with server farms. Sensors or monitoring tools normally measure the specific quantity at equidistant time instants $i\Delta$, where Δ is the reciprocal of the sampling frequency Ψ and $i \in \mathbb{Z}_{\geq 0}$. An anomaly detection procedure in the telemetry server is also regularly executed. Let ζ denote the latest time instant where the telemetry server starts an anomaly detection procedure. Any detection procedure should be finished by a certain deadline. Therefore, a detection procedure should consider only a set of L data points $\Phi(c\Delta) = \{f((c-L+1)\Delta), f((c-L+2)\Delta), \ldots, f(c\Delta)\}$, where $c\Delta$ is the latest time instant before ζ.

Definition 1 (*Intrinsic Mode Function—IMF*) If instantaneous frequency $\phi'(t) \geq 0$ and non-negative envelope $A(t) \geq 0$ vary much slower than the non-decreasing phase function $\phi(t)$, then the amplitude-modulated-frequency-modulated $u(t) = A(t)\cos(\phi(t))$ is an Intrinsic Mode Function (IMF)[6].

Definition 2 (*Intrinsic Mode Time Series—IMTS*) Let ζ denote the time instant when an anomaly detection procedure is started and $\Phi(c\Delta) = \{f((c-L+1)\Delta), f((c-L+2)\Delta), \ldots, f(c\Delta)\}$, where $c\Delta$ is the latest time instant before ζ (i.e., $c\Delta \leq \zeta < (c+1)\Delta$). If there exist K time series $\mathbf{u}_k(c\Delta) = \{u_k((c-L+1)\Delta), \ldots, u_k(c\Delta)\}$, $k = 0, \ldots, K-1$ satisfying

$$\sum_{k=0}^{K-1} \mathbf{u}_k(c\Delta) = \Phi(c\Delta), \tag{1}$$

then $\mathbf{u}_k(c\Delta)$'s are Intrinsic Mode Time Series (IMTS) components of $\Phi(c\Delta)$.

Discrete, finite-time VMD

A closed-form solution to compute Intrinsic Mode Time Series (IMTS) components $\mathbf{u}_k(c\Delta)$'s of $\Phi(c\Delta)$ does not exist. Therefore, a numerical approximation that minimises a specific error should be applied. Normally, the minimisation of $\sum_{k=0}^{K-1} \left\| \partial_t \left(\mathbf{u}_k(c\Delta) + j \cdot \mathscr{H}\mathbf{u}_k(c\Delta) \right) \cdot \mathbf{E}^T \right\|_2^2$ with a constraint defined by Equation (1) is formalised as an operation research problem[6] to determine Intrinsic Mode Functions $u_k(t)$. To include the constraint into the goal function Lagrangian multipliers are applied. That is,

$$\mathbf{u}_k(c\Delta), \Omega(c\Delta) \leftarrow \operatorname{argmin} \mathscr{L}(\mathbf{u}_k(c\Delta), \Phi(c\Delta), \lambda),$$

where $\Omega(c\Delta) = \{\omega_0(c\Delta), \ldots, \omega_{K-1}(c\Delta)\}$ is the set of centre frequencies at current timestep c corresponding to each mode,

$$\begin{aligned} \mathscr{L}(\mathbf{u}_k(c\Delta), \Phi(c\Delta), \lambda) &= \alpha \sum_{k=0}^{K-1} \left\| \partial_t \left(\mathbf{u}_k(c\Delta) + j \cdot \mathscr{H}\mathbf{u}_k(c\Delta) \right) \cdot \mathbf{E}^T \right\|_2^2 \\ &+ \left\| \Phi(c\Delta) - \sum_{k=0}^{K-1} \mathbf{u}_k(c\Delta) \right\|_2^2 + \left\langle \lambda, \Phi(c\Delta) - \sum_{k=0}^{K-1} \mathbf{u}_k(c\Delta) \right\rangle, \\ \mathbf{E} &= \left\{ e^{-j\omega_k(c\Delta)(c-L+1)\Delta}, \ldots, e^{-j\omega_k(c\Delta)c\Delta} \right\}, \end{aligned} \tag{2}$$

$j^2 = -1$, α is a pre-defined parameter, λ is the vector of Lagrangian multipliers, and $\mathscr{H}$ is the discrete Hilbert transform[37]. The Alternating Direction Method of Multipliers (ADMM)[38] could be applied to minimise $\mathscr{L}(\mathbf{u}_k(c\Delta), \Phi(c\Delta), \lambda)$. First, it iteratively computes the following quantities as

$$U_{k,n+1}(l/(L\Delta)) = \frac{F(l/(L\Delta)) - \sum_{m=0}^{k-1} U_{m,n+1}(l/(L\Delta)) - \sum_{m=k+1}^{K} U_{m,n}(l/(L\Delta)) + \Lambda_n(l/(L\Delta))/2}{1 + 2\alpha\left(l/(L\Delta) - \omega_{k,n}\right)^2}, \quad l = c - L + 1, \ldots, c,$$

$$\omega_{k,n+1} = \frac{\sum_{l=0}^{L-1}\left(l/(L\Delta) \cdot |U_{k,n+1}(l/(L\Delta))|^2/(L\Delta)\right)}{\sum_{l=0}^{L-1}\left(|U_{k,n+1}(l/(L\Delta))|^2/(L\Delta)\right)}, \quad l = c - L + 1, \ldots, c,$$

$$\Lambda_{n+1}(l/(L\Delta)) = \Lambda_n(l/(L\Delta)) + \tau\left(F(l/(L\Delta) - \sum_{k=0}^{K-1} U_{k,n+1}(l/(L\Delta))\right), \quad l = c - L + 1, \ldots, c,$$

where $F(l/(L\Delta))$ denotes the Discrete Fourier Transform (DFT) of $\Phi(c\Delta)$ as

$$F(l/(L\Delta)) = \sum_{i=0}^{L-1} f((c - L + 1 + i)\Delta) \cdot e^{-j2\pi li/(L\Delta)},$$

and $U_k(l/(L\Delta))$ is the DFT of $\{u_k((c - L + 1)\Delta), \ldots, u_k(c\Delta)\}$, $(k = 0, \ldots, K - 1)$ as

$$U_k(l/(L\Delta)) = \sum_{i=0}^{L-1} u_k((c - L + 1 + i)\Delta) \cdot e^{-j2\pi li/(L\Delta)},$$

similarly, Λ is the DFT of λ.

The algorithm stops if

$$\rho = \sum_{k=0}^{K-1} \frac{\|\mathbf{U}_{k,n+1} - \mathbf{U}_{k,n}\|_2^2}{\|\mathbf{U}_{k,n}\|_2^2} < \epsilon \ \text{ OR } \ n \geq N_{\text{iter}}, \tag{3}$$

where ϵ is the accuracy threshold and N_{iter} limits the number of VMD iterations. In the final step, the algorithm performs the Inverse Fourier Transform (IFT) to compute IMTS components.

Require: $\Phi(c\Delta)$, α, K
Ensure: $\mathbf{u}_k(c\Delta)$
  Compute the Discrete Fourier Transform (DFT) of $\Phi(c\Delta)$ and discard negative frequencies
  $\omega_{k,0} = k/(2K) \cdot 1/(L\Delta)\ \forall k = 0, \ldots, K-1$
  $\Lambda_0(l/(L\Delta)) = 0\ \forall l = c - L + 1, \ldots, c$
  $n = 0$
  repeat
    $n \leftarrow n + 1$
    for $k = 0, K - 1$ **do**
      Compute $U_{k,n+1}(l/(L\Delta))$ with parameter α
      Compute $\omega_{k,n+1}$
    end for
    Compute $\Lambda_{n+1}(l/(L\Delta))\ \forall l = c - L + 1, \ldots, c$
    Compute ρ
  until $\rho < \varepsilon$ OR $n \geq N_{\text{iter}}$
  $\mathbf{u}_k(c\Delta) \leftarrow$ Inverse Fourier Transform (IFT) of $U_k(l/(L\Delta))$

Algorithm 1. VMD $(\Phi(c\Delta), \alpha, K)$

Require: $\Phi(c\Delta)$, MIN, MAX, α, K, M_{LSTM}, M'_{LSTM}, b, WS, AP, $\widehat{\mathbf{r}}$, **AARE**

Ensure: a_c, p_c, M_{LSTM}, M'_{LSTM}, $\mathbf{r}$, $\widehat{\mathbf{r}}$, **AARE**

▷ $\Phi(c\Delta)$: measurement time series with length L
▷ $[\mathrm{MIN}, \mathrm{MAX}]$: target range of scaled values
▷ α, K: key VMD parameters (others are left as default)
▷ M_{LSTM}, M'_{LSTM}: LSTM models with input layer size b and methods $init()$, $train(data\ points)$ and $predict(data\ points)$
▷ WS: Window Size, AP: Age Power (anomaly detection parameters)
▷ $\mathbf{r}$: remainder time series
▷ $\widehat{\mathbf{r}}$, **AARE**: time series containing previous predictions and error values
▷ a_c: anomaly signal; `True` when the algorithm detects an anomaly at current timestep c
▷ p_c: pattern change signal; `True` when the algorithm detects a pattern change at current timestep c

for $i = c-L+1, c$ **do**
  $s(i) = \frac{f(i\Delta)-\min\{\Phi_c\}}{\max\{\Phi_c\}-\min\{\Phi_c\}} \cdot (\mathrm{MAX}-\mathrm{MIN}) + \mathrm{MAX}$
end for
$\sum_k \mathbf{u}_k \leftarrow \mathrm{VMD}(\mathbf{s}, \alpha, K)$
for $i = c-L+1, c$ **do**
  $r(i\Delta) = s(i\Delta) - \sum_{k=1}^{K} u_k(i\Delta)$
end for
$\widehat{r}(c\Delta) \leftarrow M_{\mathrm{LSTM}}.predict\,(r((c-b)\Delta), \ldots, r((c-1)\Delta))$
$\mathrm{AARE}(c\Delta) \leftarrow \frac{1}{\mathrm{WS}} \cdot \sum_{i=c-\mathrm{WS}+1}^{c} \left(\left(\frac{i-c}{\mathrm{WS}-1} + 1 \right)^{\mathrm{AP}} \cdot \frac{|r(i\Delta) - \widehat{r}(i\Delta)|}{r(i\Delta)} \right)$
$\mu_{\mathrm{AARE}}(c\Delta) \leftarrow \frac{1}{\mathrm{WS}} \cdot \sum_{i=c-\mathrm{WS}+1}^{c} \mathrm{AARE}(i\Delta)$
$\sigma_{\mathrm{AARE}}(c\Delta) \leftarrow \sqrt{\frac{1}{WS} \cdot \sum_{i=c-\mathrm{WS}+1}^{c} (\mathrm{AARE}(i\Delta) - \mu_{\mathrm{AARE}}(c\Delta))^2}$
$\mathrm{thd}(c\Delta) = \mu_{\mathrm{AARE}}(c\Delta) + 3 \cdot \sigma_{\mathrm{AARE}}(c\Delta)$
if $\mathrm{AARE}(c\Delta) < \mathrm{thd}(c\Delta)$ **then**
  $a_c = \mathtt{False}$ ▷ Normal operation
else
  $M'_{\mathrm{LSTM}}.init()$
  $M'_{\mathrm{LSTM}}.train\,(r((c-b)\Delta), \ldots, r((c-1)\Delta))$
  $\widehat{r}(c\Delta) \leftarrow M'_{\mathrm{LSTM}}.predict\,(r((c-b)\Delta), \ldots, r((c-1)\Delta))$
  $\mathrm{AARE}(c\Delta) \leftarrow \frac{1}{\mathrm{WS}} \cdot \sum_{i=c-\mathrm{WS}+1}^{c} \left(\left(\frac{i-c}{\mathrm{WS}-1} + 1 \right)^{\mathrm{AP}} \cdot \frac{|r(i\Delta) - \widehat{r}(i\Delta)|}{r(i\Delta)} \right)$
  if $\mathrm{AARE}(c\Delta) < \mathrm{thd}(c\Delta)$ **then**
    $a_c = \mathtt{False}$ ▷ Data pattern change
    $p_c = \mathtt{True}$
    $M_{\mathrm{LSTM}} \Leftarrow M'_{\mathrm{LSTM}}$ ▷ Copy all weights of the neural network model
  else
    $a_c = \mathtt{True}$ ▷ Anomaly
    $p_c = \mathtt{False}$
  end if
end if

Algorithm 2. AnDePeD operation

Require: c, $\Phi(c\Delta)$, MIN, MAX, α, K, M_{LSTM}, M'_{LSTM}, b, WS, AP, $\widehat{\mathbf{r}}$, **AARE**, C_{corr}, m, WS_{low}, WS_{high}, AP_{low}, AP_{high}, $\mathbf{a}$, $\mathbf{p}$, OWS, $T_{\max}$, OP, $\mathbf{o}$, δ, $\mathbf{thd}_{\text{off}}$, $\mathbf{trig}_{\text{off}}$, $\bar{D}_{\text{NORM}}$, $\bar{D}_{\text{LSTM}}$

Ensure: a_c, p_c, M_{LSTM}, M'_{LSTM}, WS, AP, $\mathbf{r}$, $\widehat{\mathbf{r}}$, **AARE**, C_{corr}, m, WS_{low}, WS_{high}, AP_{low}, AP_{high}, $\mathbf{o}$, δ, $\mathbf{a}$, $\mathbf{p}$, OP, $\mathbf{thd}_{\text{off}}$, $\mathbf{trig}_{\text{off}}$, $\bar{D}_{\text{NORM}}$, $\bar{D}_{\text{LSTM}}$

1: Record current time: $\zeta_c = \text{NOW}()$
2: $a_c, p_c, M_{\text{LSTM}}, M'_{\text{LSTM}}, \mathbf{r}, \widehat{\mathbf{r}}, \mathbf{AARE} \leftarrow \text{ANDEPED}(\mathscr{O})$
3: $a(c\Delta) = a_c$
4: $p(c\Delta) = p_c$
5: $\mathscr{O} \leftarrow \text{WS\_AP\_AUTOMATIC\_TUNING}(\mathscr{O})$
6: $\mathscr{O} \leftarrow \text{OFFSET\_COMPENSATION}(\mathscr{O})$

Algorithm 3. AnDePeD Pro operation

Design of experiments

Let us recall that the goal of our work is to develop methods that can work in a real-time environment. That is, the operation context expects the execution of algorithms at time epochs $\zeta_0, \zeta_1, \ldots, \zeta_i, \zeta_{i+1}, \ldots, \forall i \in \mathbb{Z}_{\geq 0}$. We compare our new methods, AnDePeD and AnDePeD Pro, with state-of-the-art anomaly detection algorithms (Bayesian Changepoint[20], Windowed Gaussian[21], Relative Entropy[22], KNN CAD[23], Earthgecko Skyline[24], the Alter-Re^2 algorithm[25] and AREP[26]) on traditional metrics (Precision, Recall, F-score and adjusted MCC). To obtain reproducible results, we put anomalies in data sets captured from our private server farms.

Metrics – We selected the traditional, well-established metrics of Precision, Recall and F-score (the harmonic mean of the previous two metrics)[34]. We also utilise the Matthews Correlation Coefficient (MCC), emerging as the new norm for classification metrics[34]. These values are all calculated based on the confusion matrix (true/false positives and true/false negatives), and, except for MCC, all fall in the range [0, 1]. To aid comparison, we adjusted MCC from the original $[-1, 1]$ range by adding 1 and dividing the result by 2: $\text{MCC}_{\text{adj}} = (1/2) \cdot (\text{MCC} + 1)$. However, determining the values in the confusion matrix is not as straightforward for anomaly detectors as it might be for other classification algorithms and is not done uniformly throughout the field's literature. One of the key reasons for this is the varying tolerance to the delay between an anomaly detection signal and the timestep the anomaly happened. This time offset is usually called detection delay, and we present detailed measurements below. Detection delays are usually different for each anomaly detector, each dataset, and each anomaly.

To compute the elements of the confusion matrix, we applied the approach of Lavin and Ahmad[28], where anomaly windows of length AW around the ground truth anomaly labels are used. The window's length AW is calculated by taking 10% of the dataset length and dividing this value by the number of ground truth anomalies in the dataset. This value is then rounded up to the nearest integer. Thus, we accept anomaly detections to be correct in the range $[G_i - \text{AW}, G_i + \text{AW}]$, where G_i is the ith labelled anomaly timestep. Only the first signal is counted as a true positive within the anomaly window. If there are no signals within the anomaly window, the number of false negatives increases by one. Outside the windows, every signal is regarded as a false positive, while the remaining timesteps count as true negatives. We normalise these metrics by the number AW outside the anomaly window to avoid the imbalance from only counting one TP or FN value per anomaly window. Note that we only consider the first timestep of each anomaly detection signal; thus, longer-duration signals only count as one. To sum up our evaluation process, we determine the length AW, designate anomaly windows, evaluate confusion matrix values, and then calculate the metrics Precision, Recall, F-score and adjusted MCC.

In this paper, we also present results in terms of initialisation delay and detection delay because they impact the time to take action in real-time environments. Initialisation delay is present for multiple reasons and usually differs for each algorithm. Detectors utilising a neural network to learn the typical patterns of data require several data points to predict the next value accurately. In contrast, statistically-based or probabilistic detectors usually need more historical values to achieve acceptable confidence levels in their signals. After an algorithm is past its initialisation period, it begins detecting anomalies. However, when an anomaly occurs in the data, the detector usually does not signal it immediately, that is, in the same timestep. This offset between the anomaly occurring and the detector signalling it is called detection delay. Another, more practical, yet equivalent way of defining detection delay is how many timesteps before a signal was raised the actual anomaly occurred.

Training and parameter settings

The execution of VMD and anomaly detection based on neural networks depends on VMD parameters α and K, along with hyperparameters that can be determined from data previously collected through constructing appropriate search steps in the space of parameters.

To run AnDePeD and AnDePeD Pro in real time, we chose a so-called 'supervised learning' approach which effectively meant tuning mode decomposition on historical data points that had already been collected and using either the achieved optimal VMD parameters denoted α^* and K^* or the optimal modes ($\mathbf{u}^*_{k,\text{offline}}$) themselves afterwards. Both methods assume that a satisfactory number of historical data points are available at the start of the anomaly detection process. We believe this does not limit real-world applicability significantly, as in most cases, data on the previous operation of the server farm is stored and maintained. This historical dataset must also have the same sampling period as expected in online use to maintain continuity in the periodic nature of telemetry data. We named the two operational modes Mode-I and Mode-II, and both can be split into two phases according to the 'supervised learning' paradigm: offline preparation and online use.

During offline preparation, we use the aforementioned collected dataset in both operational modes to calculate optimal VMD parameters α^* and K^* using an iterative optimisation process involving the Tree-Structured Parzen Estimator (TPESampler)[39]. However, the preparation process is slightly different. In Mode-I, we aim to achieve and store optimal VMD parameters α^* and K^*, and the objective function value to maximise is the F-score metric. A drawback of the F-score is that it necessitates the presence of labelled anomalies in the offline dataset. Nonetheless, we chose this approach for Mode-I to improve detection performance via pre-processing, which the F-score value describes effectively by one value in the range [0, 1].

For Mode II, we selected a different approach. Here, we aim to store the optimal modes achieved by VMD ($\mathbf{u}^*_{k,\text{offline}}$) themselves so that VMD only has to be run once during preparation and never during online use, significantly reducing computational resource requirements. We achieve this by selecting the Mean Square Error (MSE) between the original scaled data $\mathbf{s}$ and the sum of all modes $\sum_k \mathbf{u}_k$ as the objective function value to minimise. Similarly to Mode-I, we use TPESampler to achieve optimal α^* and K^*, but this time, we do not save them directly. Instead, we run VMD one final time with these optimal parameters and save the achieved IMTSes themselves: $\mathbf{u}^*_{k,\text{offline}}$. Using MSE means that the offline dataset for Mode-II should not contain any anomalies.

An additional requirement the data (offline and online) must conform to stems from the other component of our pre-processing procedure, namely, MinMax scaling. For that to be performed in real-time, we must possess information on the limits of the telemetry data under analysis. While this might pose some challenges regarding certain types of data generation, the data maximum and minimum can be deduced in the majority of cases (e.g., CPU utilisation falls in the range $[0\%, 100\%]$ or bytes written to a disk in one second fall between zero and the theoretical maximum write speed). In the case where no prior information is available at all, we recommend employing the work of Ogasawara et al.[40] or Passalis et al.[41]. Both articles provide adaptive normalisation algorithms for non-stationary time series data for neural network applications. Their interoperability with VMD, however, is not guaranteed. Therefore, we recommend turning the DC parameter on for VMD and reversing the order of components to the following: mode decomposition with VMD $\rightarrow$ adaptive normalisation $\rightarrow$ real-time anomaly detection.

During online use, Mode-I utilises a continuous sliding window. The algorithm maintains a circular buffer database of L_I data points; when a new data point arrives, it removes the oldest value $f((c - L_\text{I})\Delta)$ from the database and appends the new value $f(c\Delta)$ to it. This way, the latest L_I data points are always available for VMD. Each timestep, after the new data point is added to the database, we run VMD on the collected L_I data points with parameters α^* and K^* achieved from the offline preparation using TPESampler. Afterwards, the decomposed modes $\mathbf{u}_k(c\Delta)$ are removed (line 6 of Algorithm 2), and the remainder time series $\mathbf{r}$ is used for real-time anomaly detection. This approach poses high resource demands, as VMD is run every timestep, but promises high accuracy, as mode removal is based on the latest available modes $\mathbf{u}_k(c\Delta)$ possible. Notably, the length of the dataset L_I has to be sufficiently large for accurate mode decomposition to occur. This minimum length depends on the type of data, more specifically, its most significant period. Our solution for determining the VMD database length is to analyse VMD centre frequencies Ω_offline within the historically available portion of the data and calculate L_I by doubling the most significant period (calculated as the multiplicative inverse of the smallest centre frequency) as

$$L_\text{I} = 2 \cdot \left(\min_{1 \le k \le K-1} \omega_{k,\text{offline}} \right)^{-1}. \tag{4}$$

We disregard mode zero's centre frequency ω_0, since VMD usually stores significant DC components in mode zero, and as $\omega_{0,\text{offline}} \rightarrow 0 \Rightarrow 1/\omega_{0,\text{offline}} \rightarrow \infty$. We have collected all notation regarding AnDePeD and AnDePeD Pro and present it in Table 2.

Mode-II aims to conserve computational resources compared to Mode-I by performing mode decomposition only once instead of at every timestep. We run VMD at the beginning on all available historical data. The decomposed modes are saved into a database, and during online operation, the modes are continually utilised in the correct phase. A sliding window on the original decomposed modes wraps around after it reaches the end and returns to the mode's beginning. We use the points in the sliding window in our mode removal procedure as before (line 6 of Algorithm 2) and perform real-time anomaly detection on the remainder time series $\mathbf{r}$. The size of this sliding window is denoted by L_II and is left as a parameter of the algorithm.

For Mode-II, however, we have to make a list of assumptions where each item must be ensured for it to work correctly:

- The decomposed modes' length has to be an integer multiple of all periods observed in the data, but at least the most significant period present ($1/\min_{1 \le k \le K-1} \omega_{k,\text{offline}}$). If this is not the case, either end of the mode has to be truncated until the beginning of the mode can be copied after its end in the matching phase. Otherwise, we introduce out-of-phase anomalies just by this mode extension process.
- The historical data VMD is run on has to contain only normal timesteps, or extra care is to be taken to remove anomalous patterns from it in advance. This is because even though our procedure ensures that most anomalous patterns are preserved in the remainder $\mathbf{r}$, the modes $\mathbf{u}_k(c\Delta)$ themselves get significantly influenced by the presence of an anomalous peak. Suppose this point of removing anomalies from the historical data is not observed, the resulting real-time application will signal false positive anomalies in the place where real anomalies were in the historical data.
- The streaming data under analysis, extended with the historical dataset, has to be stationary. If this condition does not hold and data patterns change after some time, the decomposed modes will no longer be representative of the telemetry data, and removing them will introduce noise instead of eliminating periodicity.

This point can, however, be neglected if an algorithm is deployed that strategically reruns VMD when data patterns change, replacing the modes in the database with new ones that reflect the current state of telemetry data. We plan to formalise such an algorithm to extend our pre-processing procedure's utility.Since all three of these conditions are immediately irrelevant when using Mode-I, we recommend using Mode-II only if computational resources are scarce or the above conditions can be adequately satisfied. In those cases, however, we can use Mode-II to eliminate the most significant computational drawback of having to perform VMD every timestep; thus, the speed of our procedure will only be limited by the anomaly detection engine itself. Additionally, using Mode-II can lead to slight yet noticeable performance gains, as shown in our experiments. Furthermore, selecting Mode-I requires manual labelling of the offline dataset, presenting another detail to consider.

Table 4 summarises our parameter settings for AnDePeD and AnDePeD Pro. Scaling was utilised with parameters $[\mathrm{MIN}, \mathrm{MAX}] = [-1, 1]$. As for VMD, the selection of its parameters is somewhat more complex. Early in our experimentation, we concluded that the detection performance highly depends on two parameters of the decomposition algorithm, namely K (number of modes) and α (mode bandwidth parameter). In contrast, others had little bearing on the results and could be fixed at a given value. As for those less influential parameters, DC (whether to fix the first mode to zero frequency) is made irrelevant by our scaling approach and is therefore set to False, while the τ parameter (Lagrangian coefficient; approach to noisy data), initialisation scenario, and tolerance settings (ϵ) also did not make a significant difference as to the algorithm's performance, and were selected according to Carvalho et al.[42].

To determine the values of K^* and α^*, the Optuna optimisation framework[43] was employed. For each anomaly detection algorithm, dataset, and operational mode (I and II), we ran 50 optimisation iterations. For each iteration, Optuna calls the TPESampler method to select a new integer for K and a new real number for α from a pre-defined range: $K \in [1, 50]$ and $\alpha \in [50, 5\,000]$. These ranges were determined by early experimentation and were intentionally selected to be broad enough to accommodate a wide range of options. In general terms, TPESampler relies on independent sampling to fit a Gaussian mixture model (GMM) $l(\alpha, K)$ to the set of parameter values corresponding to the best objective values and another GMM $g(\alpha, K)$ to the remaining parameter values. TPESampler selects the parameters that maximise the ratio of the two GMMs: $\underset{\alpha,K}{\mathrm{argmax}}\{l(\alpha, K)/g(\alpha, K)\}$[44]. After value selection, Optuna pre-processes the dataset with our procedure using the selected values and runs the detector on it. Finally, it evaluates the results using our evaluation process detailed below and calculates the F-score metric for Mode-I and the MSE between $\mathbf{s}$ and $\sum_k \mathbf{u}_k$ for Mode-II as the objective function value; Mode-I aims to maximise F-score and Mode-II aims to minimise MSE. The size of the online buffer for Mode-I (L_I) is achieved via Equation (4) but is a fixed parameter for Mode-II, and was selected as $L_\mathrm{II} = 200$ data points to test the operational mode for truly resource-constrained environments.

In our experiments, the parameters for AnDePeD and AnDePeD Pro were set as shown in Table 4, Alter-Re^2 and AREP had values from their respective publications[25,26], and for the anomaly detectors from the Numenta Anomaly Benchmark (NAB)[27] corpus, we used parameter settings specified therein.

Component	Parameter	AnDePeD		AnDePeD Pro	
		Mode-I	Mode-II	Mode-I	Mode-II
Scaling	$[\mathrm{MIN}, \mathrm{MAX}]$	$[-1, 1]$			
VMD	α	Automatic			
	K	Automatic			
	τ	0.0			
	DC	False			
	init	Uniformly distributed $\omega_{k,0}$			
	ϵ	10^{-7}			
	N_iter	500			
LSTM model	b	30			
	n_neurons	30			
	n_epochs	30			
Anomaly detection	WS	1 000		Automatic (def.: 800)	
	AP	2.0		Automatic (def.: 2.5)	
Offset compensation	OWS	–		50	
	T_max	–		1 s	
Online buffer	L_I	Equation (4)	–	Equation (4)	–
	L_II	–	200	–	200

Table 4. Parameter settings for AnDePeD and AnDePeD Pro.

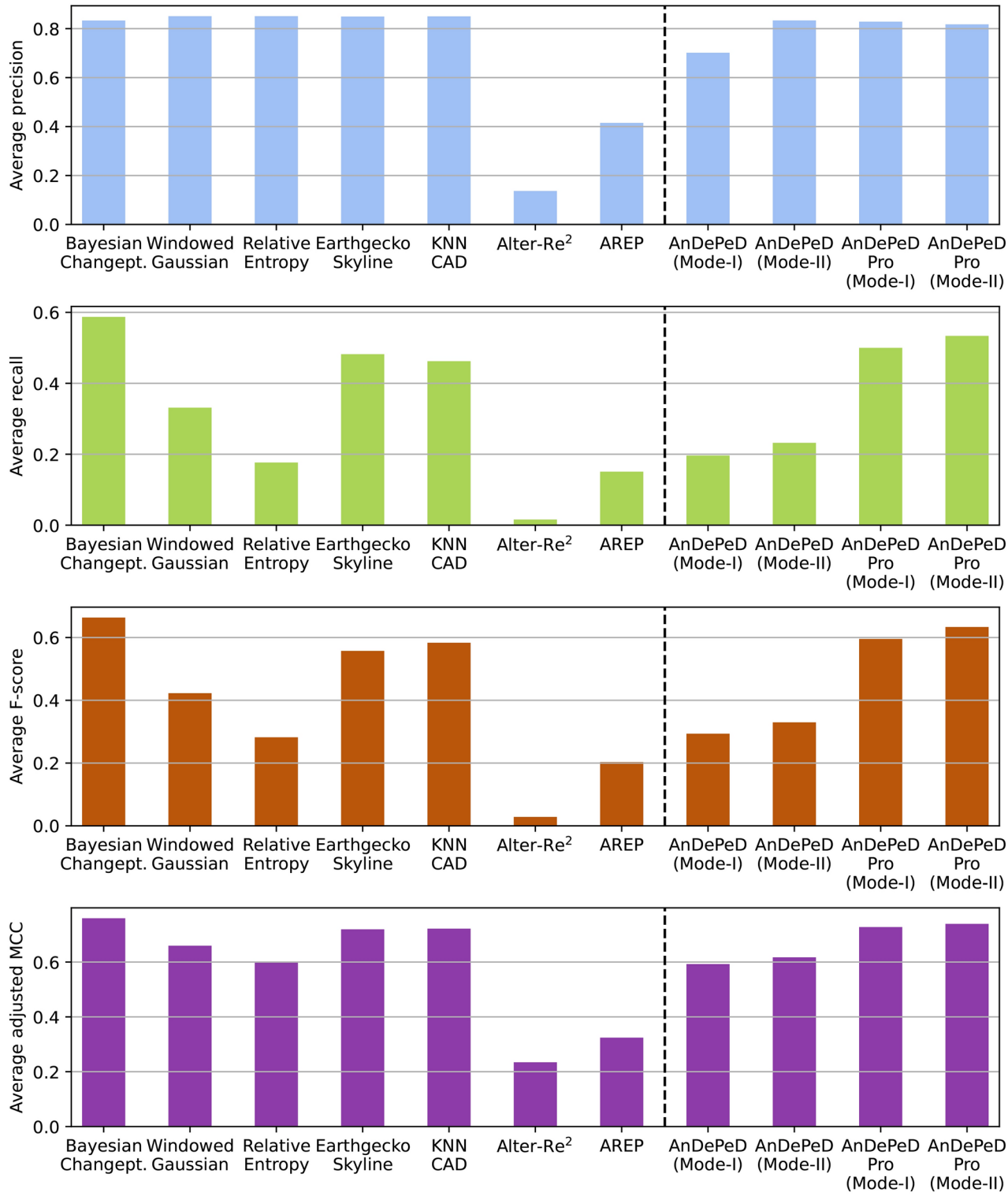


Fig. 5. Accuracy metrics.

Results

Accuracy metrics

Figure 5 plots the Precision, Recall, F-score and the Matthews correlation coefficient metrics. AnDePeD Pro achieved the second-highest average F-score metric out of all algorithms tested at 0.60 in Mode-I and 0.63 in Mode-II. First place in terms of F-score went to Bayesian Changepoint with a value of 0.66. At the same time, the following three detectors in order of performance were KNN CAD, Earthgecko Skyline and Windowed Gaussian, achieving 0.58, 0.56 and 0.42 F-score values, respectively. AnDePeD earned the next place with a

metric value of 0.29 in Mode-I and 0.33 in Mode-II, while AREP earned the penultimate position at 0.20 and Alter-Re^2 achieved the smallest average F-score at 0.03.

Besides acknowledging the utility of comparing these seven state-of-the-art anomaly detectors to each other on data they were probably never tested on, we managed to devise a detector, AnDePeD Pro, which is on par with other state-of-the-art approaches when it comes to periodic datasets in the server farm telemetry domain. We even exceeded all but one tested detectors on these four metrics by applying our new, VMD-based pre-processing method, which fulfils our aim entirely.

Initialisation delay

Our next experiment measures the delay between switching the anomaly detection algorithm on and the time it is first capable of detecting anomalies. Figure 6 presents our findings. Note that we used the number of timesteps instead of time measured in seconds. We believe it constitutes a more resource- and data-collection-independent approach, and thus provides more generally applicable results.

In the case of Bayesian Changepoint, we found that it could run from the beginning; however, as it assumes Student's t-distribution for the data under analysis, it requires a large number of points for robustness. In the implementation, Adams and MacKay used a parameter named 'maxRunLength' to initialise an array with a fixed length storing past data values and set this value as 500 timesteps. After filling this array during initialisation, Bayesian Changepoint iteratively shifts and overwrites it to contain the latest 500 values.

As for Windowed Gaussian, similar to Bayesian Changepoint, it can run from the beginning. However, being a statistical approach, it uses Gaussian probability calculation and thus requires a significant number of points for robustness. Lavin's implementation uses a so-called 'windowSize' $= 6400$ timesteps, but the more relevant parameter to initialisation delay is stepSize, which denotes the buffer length. Until this buffer is filled, Windowed Gaussian passively collects points and cannot detect anomalies. The stepSize parameter was set to 100 timesteps in the published implementation.

Relative Entropy is a windowed approach that marks each window as normal or anomalous. Being so, its initialisation requires at least an entire window (sized W) of data points to decide whether the first window is already anomalous, even though conceptually, this is somewhat unfounded. Wang et al. set the value of this parameter as $W = 52$ timesteps, and thus, somewhat favourably, we assumed the initialisation delay to be equal to that value.

Perhaps the least straightforward calculation of initialisation delay had to be performed for Earthgecko Skyline. The algorithm uses an ensemble of multiple detectors, some of which can operate right from the beginning (or after a few timesteps), while others require lengthy preparation periods. We elected to calculate the initialisation delay here as the first timestep where Earthgecko Skyline is fully operational, i.e., when all detectors begin to operate. The longest waiting period occurs in the component named 'first_hour_average' which necessitates the presence of at least one-day-old data in the current implementation. As most datasets have a sampling period of $\Delta = 5\,\mathrm{min}$, we can calculate the initialisation delay as follows: $24\,h \cdot 60\,(\mathrm{min}/h) \cdot (1/\Delta) = 288$ timesteps.

The last algorithm from the NAB corpus, KNN CAD, uses NAB's so-called 'probationaryPeriod' parameter until it passively collects data. This is the equivalent of our initialisation delay, and its value depends on the length of the dataset (L) and the 'probationPercent' (PP) parameters. Assuming an accurate average data length of $L = 5000$ and using Burnaev and Ishimtsev's selection of $\mathrm{PP} = 15\%$, we arrive at the initialisation delay being equal to $L \cdot \mathrm{PP} = 750$ timesteps.

Alter-Re^2, AREP, AnDePeD and AnDePeD Pro inherit their initialisation delay from ReRe[32]. This delay only depends on the algorithms' look-back parameter, b. The preparation period can be split into three steps:

1. The algorithm passively collects data points (while the current timestep $c < b - 1$);

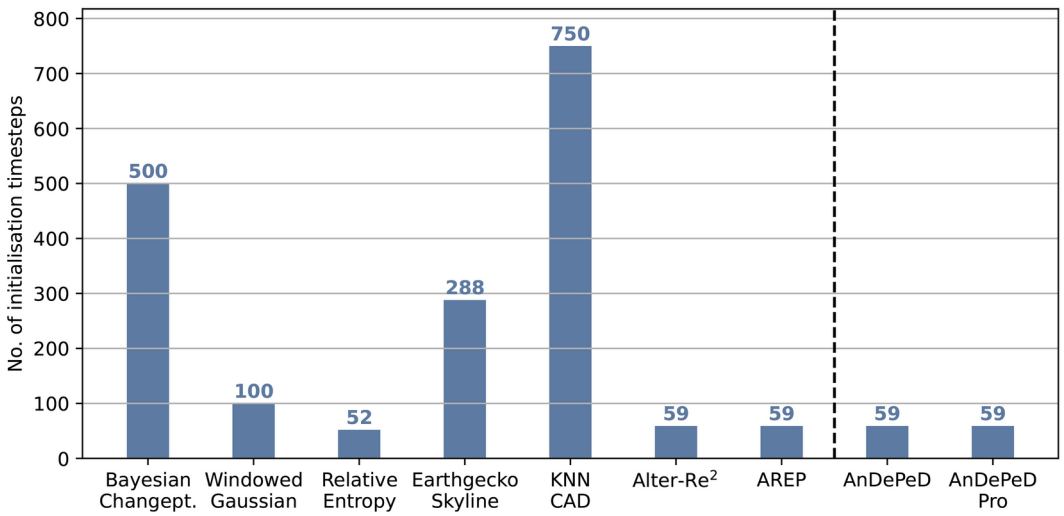


Fig. 6. Performance comparison results on initialisation delay.

2. For one timestep, the algorithm trains its LSTM models and uses that to predict the next timestep (at $c = b - 1$);
3. Besides training and prediction, it also calculates the $\mathrm{AARE}(c\Delta)$ error values (while $b - 1 < c < 2b - 1$). Consequently, the initialisation delay for Alter-Re^2, AREP, AnDePeD and AnDePeD Pro is $2b - 1$ timesteps long. Given the setting of $b = 30$ in our experiments shown in Table 4 for all of these detectors, this evaluates to 59 timesteps.

Regarding initialisation delay, Alter-Re^2, AREP, AnDePeD and AnDePeD Pro are in second place and safely among the fastest detectors tested in achieving the ready-to-use state. Our AnDePeD Pro is a reasonable trade-off between fast initialisation and accurate prediction performance.

Detection delay

Detection delay is $S_i - G_i$, where G_i is the time instant of anomaly i and S_i is the signalling timestep. Also, Doshi et al.[45] define average detection delay ADD as

$$\mathrm{ADD} = \frac{1}{N_{\mathrm{anom}}} \cdot \sum_{i=0}^{N_{\mathrm{anom}}} (S_i - G_i),$$

where N_{anom} is the total number of anomalies. It is worth emphasising that the possible outcomes of all the available algorithms are as follows:

- True positive signal within the anomaly window (size AW) and after G_i: detection delay is $S_i - G_i$, as discussed.
- No signal within the anomaly window after G_i (so-called false negative): we assign the size of the anomaly window to the detection delay, $S_i - G_i := \mathrm{AW}$.Figure 7 presents the average number of steps until detection (i.e., detection delay), the minimum and the maximum detection delay of the algorithms. Note that signals within the anomaly window before the ground truth timesteps are not considered. Additionally, in the case of our detectors (AnDePeD and AnDePeD Pro), the pre-processing results were achieved in Mode-I (we experienced only slight changes in Mode-II). At first glance, the number of timesteps the algorithms presented might seem high due to a choice to handle false negatives by assigning the anomaly window length (AW) because signalling is only valuable if an anomaly can be detected after a specific threshold. As shown in Fig. 7, AnDePeD Pro achieved the lowest average detection delay out of all detectors tested, 80 timesteps precisely. Bayesian Changepoint and Earthgecko Skyline follow AnDePeD Pro with average values of 93 timesteps and 100 timesteps. The former had, on average, 16% longer delays until signalling the anomaly. AnDePeD beat Alter-Re^2 with a delay of 146 timesteps compared to 160. Our latest detector, AnDePeD Pro, initialises relatively fast and has the lowest delay measured when detecting anomalies. The reduction in detection delay is highly likely the result of removing intrinsic mode time series from the original time series.

Complexity

Each algorithm uses n_{p} previous data points at the execution time instant. The complexity of the algorithms is summarised in Table 5. We have performed an extensive numerical study to determine required data points (n_{p}) so all the algorithms could produce similar accuracy. The result is summarised in Table 6.

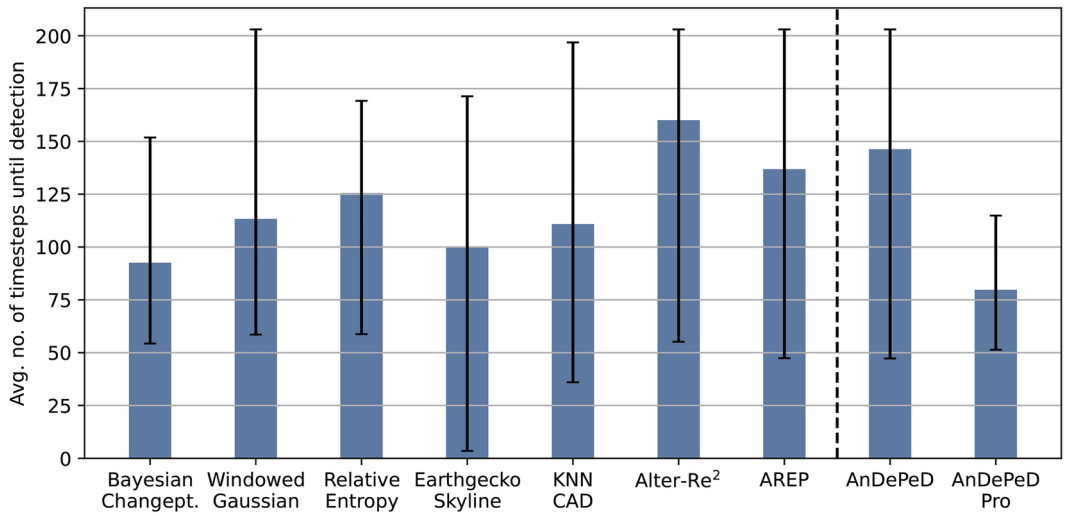


Fig. 7. Performance comparison results on detection delay.

Anomaly detector	Algorithm complexity
Bayesian Changepoint	$O(n_\mathrm{p})$
Windowed Gaussian	$O(n_\mathrm{p})$
Relative Entropy	$O(n_\mathrm{p})$
Earthgecko Skyline	$O(n_\mathrm{p})$
KNN CAD	$O(n_\mathrm{p}^3)$
Alter-Re^2	$O(n_\mathrm{p})$
AREP	$O(n_\mathrm{p}^2)$
AnDePeD	$O(n_\mathrm{p} \cdot \log(n_\mathrm{p}))$
AnDePeD Pro	$O(n_\mathrm{p}^2)$

Table 5. Computational complexity of algorithms tested by the number of past data points - n_p.

Anomaly detector	n_p
Bayesian Changepoint	500
Windowed Gaussian	$6\,400$
Relative Entropy	52
Earthgecko Skyline	no limit
KNN CAD	$1\,500$
Alter-Re^2	$1\,000$
AREP	auto (initial: 800)
AnDePeD (Mode-I)	Equation (4)
AnDePeD Pro (Mode-II)	200

Table 6. Maximum number of past data points used.

Bayesian Changepoint uses the same number of past data points as its initialisation delay, as the variable 'maxRunLength' determines the size of an array storing past values, which are shifted and overwritten every timestep. Until the array is filled, Bayesian Changepoint passively collects data points. Windowed Gaussian, being a statistical approach, requires many points for robustness. The implementation utilises a 'windowSize'-long array to store past data points, which Lavin has set to 6400. As mentioned, Relative Entropy is a windowed approach. It collects a fixed number of data points and then evaluates whether the window is anomalous. Relative Entropy is implemented so that it stores all data points but uses only the latest W values, where the window size is set to $W = 52$. KNN CAD stores all data points in a buffer with no theoretical limit on its size. However, KNN CAD only uses the latest $2 \cdot \mathrm{len} \cdot \mathrm{PP}$ points for training, where len is the total length of data loaded and PP is the preparation period parameter. For the former, we chose the charitable $\mathrm{len} = 5\,000$, as the average length of datasets is even higher, while for the latter, we used the $\mathrm{PP} = 15\%$ setting from the KNN CAD algorithm code. In Alter-Re^2, as in many previous approaches, the window size parameter determines the number of past data points we set to $\mathrm{WS} = 1000$. AREP contains the automatic tuning of WS, the number of past data points used is no longer constant. Based on extensive experimentation, we set its initial value to 800. As for AnDePeD and AnDePeD Pro, the variable L (L_I for Mode-I and L_II for Mode-II) provides an upper limit on the number of past data points used. In the first operational mode, double of the most considerable VMD mode period is assigned to L_I as shown in Equation (4), while we left L_II as a fixed parameter and set it to 200.

From Tables 5 and 6, we can conclude that even though AnDePeD and especially AnDePeD Pro are computationally complex and resource intensive when choosing Mode-II, they require fewer points than any other approach except for Relative Entropy. The highest order algorithm, KNN CAD, also requires the second highest n_p, on the other hand, Bayesian Changepoint achieves linear complexity and uses the third lowest number of data points, representing another valid choice.

Discussion

When considering our experimental results on the three measurements simultaneously, selecting the optimal detector depends on a specific use case. For example, if detection speed is of the essence, and false positives or missed anomalies are not an issue, more weight must be given to delay results. Conversely, for applications where precision and accuracy matter more than how fast the algorithm raises the signal, the focus of a study should be performance results. From the numerical results, AnDePed Pro is the best because it achieves a trade-off between the performance and the detection delay.

Additionally, regarding the number of past data points required, AnDePeD and AnDePeD Pro in Mode-II achieved second place after Relative Entropy, leading to a conservative use of system memory. Results showed that the maximum detection delay of the AnDePeD Pro is the smallest, which justifies the goal of the paper.

When implementing AnDePeD or AnDePeD Pro in a server farm operations centre, there are a few aspects to consider. Suppose the telemetry system is built from the ground up, we recommend the OpenTelemetry (OTel)[33] framework with distributed tracing and metrics collection capabilities. Moreover, it enables real-time data transmission while maintaining the quality of service and reducing the impact on the performance of the monitored systems. One can easily integrate AnDePeD and AnDePeD Pro into OTel, which also supports many TSDBs for storing collected data.

If a telemetry system is already deployed, AnDePeD and AnDePeD Pro can be included as an external module that receives telemetry data points and returns anomaly signals. As AnDePeD and AnDePeD Pro are univariate detectors, each measured metric needs to be analysed individually, and the number of concurrently measured metrics determines the amount of resources required. Therefore, we recommend selecting only the metrics with the highest potential for capturing anomalous behaviour. In large server farms, aggregating multiple metrics into one and effectively reducing data dimensions is an apt approach for scalability, and we plan to provide direct solutions for that in the future.

Our experiments presented above were run on periodic datasets to assess the performance of AnDePeD and AnDePeD Pro in comparison to the state-of-the-art. Although we did not see large performance drops during preliminary experiments on aperiodic data with our latest detectors, our aim was not to create all-around suitable algorithms. We already have appropriate solutions for aperiodic datasets in the form of Alter-Re^2 and AREP; we recommend using them or other more suitable algorithms, which do not unnecessarily introduce the pre-processing step using VMD, for aperiodic data in non-seasonal environments.

Limitations

Computational complexity is higher for AnDePeD and especially AnDePeD Pro than most other approaches except for KNN CAD and AREP. This poses a challenge for environments where CPU power is scarce, even though that rarely describes the operations centre of a server farm. In our experiments, AnDePeD gives the anomaly signal in a maximum of a few seconds on commercial off-the-shelf hardware, which far exceeds the five-minute measurement period. Furthermore, as discussed, training the LSTM model is the longest step in practice, which is independent of the number of past data points. Thus, the relevance of computational complexity is limited for detectors using neural network models.

Other limitations include that AnDePeD and AnDePeD Pro rely on collected past data points to calculate optimal VMD parameters α^* and K^* in Mode-I or the optimal modes ($\mathbf{u}^*_{k,\mathrm{offline}}$) themselves in Mode-II. This poses a demand for system operators to store a higher number of data points before use. Additionally, operators have to label anomalies for Mode-I or ensure none exist in the data for Mode-II. For the latter operational mode, even stricter demands are placed on the data, outlined in the 'Training and parameter settings' subsection above.

Following the best-of-practice approach used in the literature, we compared several datasets (from the literature and our server farm) with artificial ground truths. However, this work and other literature papers lack a systematic experimentation and evaluation strategy for anomalies that may happen with a very small probability in server farms. One should also estimate the confidence interval of decisions on whether an anomaly occurs at specific time epochs. Since data collected from servers are more or less correlated due to resource allocation decisions in managing a server farm, developing algorithms that deal with multi-dimensional data is unavoidable. The development of a new evaluation strategy is part of our future plan.

Future scope

We intend to test our algorithms with other, longer measured time series to assess whether the results presented in this article are the consequence of our datasets under examination. Extending tests to other data sources from within the server farm telemetry domain and beyond is in the scope of our interests and plans.

Another unexplored issue is whether patterns or data classes can be identified with similar optimal VMD parameters. If achieved, pre-processing parameter selection could be aided by a classification procedure that could select VMD parameters automatically, negating the need to run optimisation on the type of data under analysis. For the online operation of our procedure, we would like to automatise cleaning-up modes to remove non-regular behaviour and identify when to re-tune VMD parameters, thus enabling adaptation to changing data patterns. In terms of optimisation, we would also like to employ different sampling methods besides TPESampler and add pruning to speed up the optimisation of VMD parameters.

The performance and time delay experiment results outlined above can presumably be improved further by more experimentation. We will also extend our scope regarding pre-processing and transformation methods, datasets, data types, anomaly detectors, and data dimensions.

Conclusions

We have presented a novel anomaly detection method, AnDePeD, capable of dealing with periodic data patterns and its improved version, AnDePeD Pro. We have shown that removing intrinsic mode time series from the scaled original data improves the performance of our algorithms. Evaluating our real-time pre-processing approach on seven periodic datasets that match our use case of server farm telemetry, we found that AnDePeD Pro achieved second place in performance among modern, well-established detectors. Our initialisation delay and detection delay experiments solidify AnDePeD Pro as the best detector tested since it was not only among the best in terms of performance but also achieved first place in detection delay, signalling anomalies the fastest of all tested methods. Regarding initialisation, although not the fastest, it shared the second place among the best-performing approaches. Our anomaly detectors, AnDePeD and AnDePeD Pro, provide an apt solution to

real-time anomaly detection in the server farm telemetry domain. We believe our algorithms can also be applied to other real-time environments.

As part of our future plans, we aim to extend our research to multi-dimensional time series datasets. Considering the correlation between different types of information from the same data source (e.g., the temperature, CPU and port information of the same router) should lead to better performance than one-dimensional analysis on all datasets separately. The nature of periodic and aperiodic time series may require corresponding mode decomposition and transformation algorithms. We will create a framework to select a suitable data transformation approach.

Supplementary information

We include the pseudocode of the automatic tuning and offset compensation components (Algorithm S1 and Algorithm S2) in the attached Supplementary Information document, along with additional preliminary results on data from our sever farm telemetry testbench (Figure S1).

Data availability

We put all datasets along with codes, an instruction to run experiments and results at the following address https://github.com/VDanielL/AnDePeD.

Received: 10 July 2024; Accepted: 12 September 2024
Published online: 07 October 2024

References

1. Samdanis, K. & Taleb, T. The road beyond 5G: A vision and insight of the key technologies. *IEEE Network* **34**, 135–141. https://doi.org/10.1109/MNET.001.1900228 (2020).
2. Gui, G., Liu, M., Tang, F., Kato, N. & Adachi, F. 6G: Opening new horizons for integration of comfort, security, and intelligence. *IEEE Wirel. Commun.* **27**, 126–132. https://doi.org/10.1109/MWC.001.1900516 (2020).
3. Viswanathan, H. & Mogensen, P. E. Communications in the 6G era. *IEEE Access* **8**, 57063–57074. https://doi.org/10.1109/ACCESS.2020.2981745 (2020).
4. Ziegler, V. *et al.* 6G architecture to connect the worlds. *IEEE Access* **8**, 173508–173520. https://doi.org/10.1109/ACCESS.2020.3025032 (2020).
5. Bertenyi, B. 5G evolution: What's next?. *IEEE Wirel. Commun.* **28**, 4–8. https://doi.org/10.1109/MWC.2021.9363048 (2021).
6. Dragomiretskiy, K. & Zosso, D. Variational mode decomposition. *IEEE Trans. Signal Process.* **62**, 531–544. https://doi.org/10.1109/TSP.2013.2288675 (2014).
7. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* **9**, 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735 (1997).
8. Zouari, F., Ibeas, A., Boulkroune, A., Cao, J. & Mehdi Arefi, M. Adaptive neural output-feedback control for nonstrict-feedback time-delay fractional-order systems with output constraints and actuator nonlinearities. *Neural Netw.* **105**, 256–276, (2018).
9. Nassif, A. B., Talib, M. A., Nasir, Q. & Dakalbab, F. M. Machine learning for anomaly detection: A systematic review. *IEEE Access* **9**, 78658–78700. https://doi.org/10.1109/ACCESS.2021.3083060 (2021).
10. Pang, G., Shen, C., Cao, L. & Hengel, A. V. D. Deep learning for anomaly detection: A review. *ACM Comput. Surv.*[SPACE]https://doi.org/10.1145/3439950 (2021).
11. Friha, O. *et al.* Felids: Federated learning-based intrusion detection system for agricultural internet of things. *J. Parallel Distributed Comput.* **165**, 17–31. https://doi.org/10.1016/j.jpdc.2022.03.003 (2022).
12. Schmidl, S., Wenig, P. & Papenbrock, T. Anomaly detection in time series: A comprehensive evaluation. *Proc. VLDB Endow.* **15**, 1779–1797 (2022).
13. Han, S., Hu, X., Huang, H., Jiang, M. & Zhao, Y. Adbench: Anomaly detection benchmark. In Koyejo, S. *et al.* (eds.) *Advances in Neural Information Processing Systems*, vol. 35, 32142–32159 (Curran Associates, Inc., 2022).
14. Liu, B. *et al.* A deep learning framework assisted echocardiography with diagnosis, lesion localization, phenogrouping heterogeneous disease, and anomaly detection. *Sci. Rep.* **13**(3), 2023. https://doi.org/10.1038/s41598-022-27211-w (2023).
15. Tian, Z., Zhuo, M., Liu, L., Chen, J. & Zhou, S. Unsupervised anomaly detection with generative adversarial networks in mammography. *Sci. Rep.* **13**(2925), 2023. https://doi.org/10.1038/s41598-023-29521-z (2023).
16. Tian, Z., Zhuo, M., Liu, L., Chen, J. & Zhou, S. Anomaly detection using spatial and temporal information in multivariate time series. *Sci. Rep.* **13**(4400), 2023. https://doi.org/10.1038/s41598-023-31193-8 (2023).
17. Shi, H., Guo, J., Deng, Y. & Qin, Z. Machine learning-based anomaly detection of groundwater microdynamics: Case study of Chengdu, China. *Sci. Rep.* **13**(14718), 2023. https://doi.org/10.1038/s41598-023-38447-5 (2023).
18. Liu, J. *et al.* Deep industrial image anomaly detection: A survey. *Mach. Intell. Res* **21**, 104–135. https://doi.org/10.1007/s11633-023-1459-z (2024).
19. Hand, D. J. Classifier technology and the illusion of progress. *Stat. Sci.*[SPACE]https://doi.org/10.1214/088342306000000060 (2006).
20. Adams, R. P. & MacKay, D. J. C. Bayesian Online Changepoint Detection (2007). arXiv:0710.3742.
21. Lavin, A. Windowed Gaussian [Online code repository]. https://github.com/numenta/NAB/blob/master/nab/detectors/gaussian (2016). Accessed: 2024-06-24.
22. Wang, C. *et al.* Statistical techniques for online anomaly detection in data centers. In *12th IFIP/IEEE International symposium on integrated network management (IM 2011) and Workshops*, 385–392, https://doi.org/10.1109/INM.2011.5990537 (2011).
23. Burnaev, E. & Ishimtsev, V. Conformalized density-and distance-based anomaly detection in time-series data, https://doi.org/10.48550/ARXIV.1608.04585 (2016).
24. Stanway, A. & Wilson, G. Earthgecko Skyline [Online code repository]. https://github.com/earthgecko/skyline (2023). Accessed: 2024-06-24.
25. Vajda, D., Pekar, A. & Farkas, K. Towards Machine Learning-based Anomaly Detection on Time-Series Data. *Infocommun. J.*[SPACE]https://doi.org/10.36244/ICJ.2021.1.5 (2021).
26. Farkas, K. AREP: An adaptive, machine learning-based algorithm for real-time anomaly detection on network telemetry data. *Neural Comput. Appl.* **35**, 6079–6094. https://doi.org/10.1007/s00521-022-08000-y (2023).
27. Numenta Inc. NAB: Numenta Anomaly Benchmark [Online code repository]. https://github.com/numenta/NAB (2022). Accessed: 2024-06-24.
28. Lavin, A. & Ahmad, S. Evaluating Real-Time Anomaly Detection Algorithms – The Numenta Anomaly Benchmark. In *Proceedings of the IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, 38–44, https://doi.org/10.1109/ICMLA.2015.141 (2015).

29. Stanway, A. & Wilson, G. Earthgecko Skyline [Online documentation]. https://earthgecko-skyline.readthedocs.io/en/latest/overview.html (2023). Accessed: 2024-08-05.
30. Lee, T. J., Gottschlich, J., Tatbul, N., Metcalf, E. & Zdonik, S. Greenhouse: A Zero-Positive Machine Learning System for Time-Series Anomaly Detection (2018). arXiv:1801.03168.
31. Lee, M.-C., Lin, J.-C. & Gran, E. G. RePAD: Real-Time Proactive Anomaly Detection for Time Series. In Barolli, L., Amato, F., Moscato, F., Enokido, T. & Takizawa, M. (eds.) *Proceedings of the Advanced Information Networking and Applications*, 1291–1302 (Springer International Publishing, Cham, 2020).
32. Lee, M.-C., Lin, J.-C. & Gran, E. G. ReRe: A Lightweight Real-Time Ready-to-Go Anomaly Detection Approach for Time Series. In *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, 322–327, https://doi.org/10.1109/COMPSAC48688.2020.0-226 (2020).
33. Blanco, D. G. *Practical OpenTelemetry* (Apress, 2023).
34. Chicco, D. & Jurman, G. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*[SPACE]https://doi.org/10.1186/s12864-019-6413-7 (2020).
35. Barlacchi, G. *et al.* A multi-source dataset of urban life in the city of Milan and the province of Trentino. *Sci. Data* **2**, 150055. https://doi.org/10.1038/sdata.2015.55 (2015).
36. Wittig, A. & Wittig, M. *Amazon Web Services in Action: An in-depth guide to AWS* (Simon and Schuster, 2023).
37. Kak, S. The discrete Hilbert transform. *Proc. IEEE* **58**, 585–586. https://doi.org/10.1109/proc.1970.7696 (1970).
38. Glowinski, R. *On Alternating Direction Methods of Multipliers: A Historical Perspective, 59–82* (Springer, Netherlands, Dordrecht, 2014).
39. Bergstra, J., Bardenet, R., Bengio, Y. & Kégl, B. Algorithms for hyper-parameter optimization. In Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F. & Weinberger, K. (eds.) *Advances in Neural Information Processing Systems*, vol. 24 (Curran Associates, Inc., 2011).
40. Ogasawara, E. *et al.* Adaptive normalization: A novel data normalization approach for non-stationary time series. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, 1–8, https://doi.org/10.1109/IJCNN.2010.5596746 (2010).
41. Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M. & Iosifidis, A. Deep adaptive input normalization for time series forecasting. *IEEE Trans. Neural Netw. Learning Syst.* **31**, 3760–3765. https://doi.org/10.1109/TNNLS.2019.2944933 (2020).
42. Carvalho, V. R., Moraes, M. F., Braga, A. P. & Mendes, E. M. Evaluating five different adaptive decomposition methods for EEG signal seizure detection and classification. *Biomed. Signal Process. Control* **62**, 102073. https://doi.org/10.1016/j.bspc.2020.102073 (2020).
43. Akiba, T., Sano, S., Yanase, T., Ohta, T. & Koyama, M. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, 2623–2631, https://doi.org/10.1145/3292500.3330701 (Association for Computing Machinery, New York, NY, USA, 2019).
44. Watanabe, S. Tree-Structured Parzen Estimator: Understanding Its Algorithm Components and Their Roles for Better Empirical Performance (2023). arXiv:2304.11127.
45. Doshi, K., Abudalou, S. & Yilmaz, Y. Reward once, penalize once: Rectifying time series anomaly detection. In *2022 International Joint Conference on Neural Networks (IJCNN)*, 1–8, https://doi.org/10.1109/IJCNN55064.2022.9891913 (2022).

Acknowledgements

Dániel László Vajda has been partially supported by the ÚNKP-23-3-I-BME-21 New National Excellence Programme granted by the Ministry of Culture and Innovation (KIM) of Hungary via the National Research, Development and Innovation Office (NKFI Alap). This work has been partially supported by the Hungarian Scientific Research Fund OTKA K-138208 project.

Author contributions

D.L.V. performed the necessary literature review; took a leader role in devising AnDePeD; designed, implemented, ran and evaluated the experiments; and participated in the manuscript writing. T.V.D. supervised the mathematical formulation of the algorithms, equations and derivations, and participated in the manuscript writing. T.B. participated in the manuscript writing. K.F. took a leader role in devising AnDePeD Pro and participated in the manuscript writing. All authors reviewed the manuscript.

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-024-72982-z.

Correspondence and requests for materials should be addressed to D.L.V.