Applied Soft Computing Journal 108 (2021) 107443

Contents lists available at ScienceDirect

Applied Soft Computing Journal

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

Unsupervised anomaly detection with LSTM autoencoders using statistical data-filtering

Sepehr Maleki[a,*], Sasan Maleki[b], Nicholas R. Jennings[c]

[a] *University of Lincoln, United Kingdom*
[b] *Sharif University of Technology, Iran*
[c] *Imperial College London, United Kingdom*

A R T I C L E I N F O

Article history:
Received 24 August 2020
Received in revised form 13 April 2021
Accepted 16 April 2021
Available online 24 April 2021

Keywords:
LSTM networks
Autoencoders
Anomaly detection
Changepoint detection
IoT

A B S T R A C T

To address one of the most challenging industry problems, we develop an enhanced training algorithm for anomaly detection in unlabelled sequential data such as time-series. We propose the outputs of a well-designed system are drawn from an unknown probability distribution, $\mathcal{U}$, in normal conditions. We introduce a probability criterion based on the classical central limit theorem that allows evaluation of the likelihood that a data-point is drawn from $\mathcal{U}$. This enables the labelling of the data on the fly. Non-anomalous data is passed to train a deep Long Short-Term Memory (LSTM) autoencoder that distinguishes anomalies when the reconstruction error exceeds a threshold. To illustrate our algorithm's efficacy, we consider two real industrial case studies where gradually-developing and abrupt anomalies occur. Moreover, we compare our algorithm's performance with four of the recent and widely used algorithms in the domain. We show that our algorithm achieves considerably better results in that it timely detects anomalies while others either miss or lag in doing so.

© 2021 Elsevier B.V. All rights reserved.

1. Introduction

Anomaly detection is the task of discovering data patterns whose characteristics statistically differ from those available during training (regarded as normal). Several challenges make anomaly detection a formidable task in practice. In general, it is not easy to characterise every possible normal behaviour, and therefore, distinguishing the abnormal behaviour (especially on the boundaries) can become difficult. Moreover, in many areas, the characteristic notion of normal behaviour evolves through time, and those currently regarded as normal might represent anomalies in the future. For example, jet engines go through several modes during their operations that result in similar characteristics as anomalies. Additionally, anomalous data characteristics differ from one domain to another, which makes the extension not easy. Due to such difficulties, most of the current anomaly detection methods solve a domain-specific formulation of the problem.

Anomaly detection in time-series is one of the main challenges in today's industry, where an unprecedented number of sensors are utilised to monitor various processes. Consequently, extensive research has been carried out to develop intelligent agents to solve the problem and facilitate remote monitoring challenges. These agents often rely on algorithms that require offline training on "clean" or labelled data, which is costly and labour-intensive. Moreover, in general, these agents are not designed to adapt easily from one domain to another. Additionally, it is often required to trade off between accuracy and false-alarm rates — especially when dealing with complex systems with several operation modes [1–3].

1.1. *Related work*

A comprehensive review of the anomaly detection problem and current limitations are given in [2]. Here, we give an overview of common supervised and unsupervised techniques for anomaly detection and briefly discuss them.

1.1.1. *Supervised algorithms*

Many of the supervised algorithms are based on one-class classification using Support Vector Machines (SVMs). In the training process, SVMs use kernels (e.g., radial basis functions) to learn complex boundaries containing training data instances. For each test instance, SVM is used to determine whether the instance falls within the learned region. Instances that fall outside these regions are called anomalous. Examples of such algorithms include [4,5].

Model-based methods are another approach to anomaly detection. Specifically, these models are sought after in industries where a system's behaviour can be described via a mathematical model. Some of these approaches include observer-based [6,7],

* Corresponding author.
E-mail addresses: smaleki@lincoln.ac.uk (S. Maleki), smaleki@ce.sharif.edu (S. Maleki), n.jennings@imperial.ac.uk (N.R. Jennings).

https://doi.org/10.1016/j.asoc.2021.107443
1568-4946/© 2021 Elsevier B.V. All rights reserved.

parity-space-based [8], and parameter identification-based methods [9]. The main challenge in this approach is the development difficulty encountered when dealing with complex non-linear systems.

Although in recent years, deep neural networks have been mainly used to develop unsupervised algorithms [10,11], they are also used to develop supervised anomaly detection algorithms for one-class, as well as multi-class settings [12]. The application of these networks typically consists of two stages. First, a neural network is trained on previously labelled "normal" data to learn a latent representation that describes the expected behaviour. The model is then fed with test data that can contain anomalies that are detected if they are rejected by the model [13].

Supervised anomaly detection techniques are, in general, very accurate and straightforward to optimise. They also offer the freedom to influence prior knowledge in the design. However, the significant difficulty in developing these algorithms concerns the scarcity of the labelled data. Additionally, the training procedure naturally prevents them from online anomaly detection.

1.1.2. Unsupervised algorithms

Despite the increasing availability of data, the lack of appropriate training examples remains one of the major challenges for anomaly detection. Consequently, in applications where the anomaly is rare but can be catastrophic, unsupervised learning schemes are often the better choice.

A common class of these algorithms are known as changepoint detection (see [14] for a detailed survey), where changepoints are defined as abrupt variations in the generative parameters of a data sequence. These techniques mainly rely on inferences drawn from the statistical analysis of data. A well-established example is the Bayesian online changepoint detection algorithm [15], where a sequence of observations $x_1, x_2, \ldots, x_T$ may be divided into non-overlapping product partitions, and the data within each partition are independently and identically sampled from some probability distribution. Then, the probability distribution of the length of the current "run" since the last changepoint is computed, and it is inferred whether the current point is a changepoint. Although the algorithm is highly modular and can be applied to various data, the probability distribution of the complete run lengths is computed repeatedly. Therefore, as the length, n, increases, it becomes more computationally expensive (i.e., $\mathcal{O}(n^2)$) to infer changepoints. This is especially significant in industries where data is generated fast and in high volumes.

Another example of changepoint algorithms is [16], where two detectors are developed to analyse the data streams' statistical properties, and a changepoint is detected when both detectors find a significantly different pattern from those available in training. Although this method can be employed as an online algorithm, the training process is offline.

Robust Principal Component Analysis (RPCA) [17] is another method that shown successful anomaly detection. The method proposes that the data, X, can be divided into two parts such that $X = L_D + S$, where L_D, the true low-rank component, can be effectively reconstructed, and the sparse matrix S is the part that contains anomalies. However, these implementations of PCA normally require a thorough examination of the full dataset and are not suitable for scenarios where future data is not available.

Symbolic Aggregate Approximation (SAX) [18] is another example of unsupervised algorithms where the data is split into segments. Then a symbolic representation for each segment is generated that allows for dimensionality reduction and novelty detection. However, a significant shortcoming of SAX is that it only reflects the segment mean values and does not consider the temporal information (e.g., the trend of a change). Moreover, similar to RPCA and RDA, it requires examining the full data-set a priori to generate symbols.

Autoencoders are another unsupervised learning technique where neural networks are leveraged to learn latent representations of the data. These architectures have gained much attraction for anomaly detection [19–21]. They are designed to reconstruct the input at the output, where the reconstruction error is used as a metric to detect anomalous data. An interesting variation of these architectures is Long Short Term Memory (LSTM) networks used for the encoding and decoding units. LSTM networks are recurrent neural networks that have shown state-of-the-art performance in sequence learning tasks, including anomaly detection [22,23]. An LSTM-based encoder maps the input sequence to a latent vector representation of fixed length. Then, the decoder, another LSTM network, uses the latent representation to reconstruct the input sequence. LSTM-based autoencoders achieve even better results in anomaly detection compared to autoencoders [24–26]. However, significant improvements in detection accuracy can only be achieved using complex LSTM networks with large memory requirements and high computational complexity [27]. Therefore, efficient hardware architectures and compression schemes for LSTM networks have been proposed [27–30] to meet the desired low-latency and energy-efficiency requirements of the real-world applications. For example, [31] uses LSTM networks to monitor the LHC superconducting magnets at CERN despite the very low time resolution (one sample for 400 ms) of the data. Nonetheless, a major limitation of autoencoders is that the noise and outliers within the input data impair the performance of the algorithms [2,4].

In summary, in industrial settings, existing methods suffer from one or more of the following limitations [2,4]:

(a) lack of robustness against noise (e.g., [23]);
(b) inability to perform online anomaly detection (e.g., [18]).
(c) non-linear computational complexity as more data is observed (e.g., [15]);

Due to the scarcity of labelled datasets, these limitations and challenges hinder the otherwise strong performance of LSTM autoencoders for anomaly detection. Therefore, an enhanced training algorithm is required which has the following desired properties: (i) it enables unsupervised labelling of data to reduce observation noise, (ii) it can be applied for both online and offline, (iii) it is memory-efficient so that it is suitable for big data, (iv) it can be configured to trade off the false-alarm rate for accuracy.

Against this background, this paper proposes an algorithm that addresses the limitations of the existing works and exhibits the properties above. In particular, this algorithm improves the performance of LSTM-based autoencoders and especially suits the cases where the unlabelled training data consists of incidents or patterns that are characteristically similar to anomalies. In more detail, it achieves these by:

(a) implementing a novel training technique that reduces the amount of noise and outliers in the training data. This feature, as shown in Section 4, achieves considerably better results in terms of time and accuracy compared to other LSTM, LSTM autoencoder, and autoencoder algorithms [23,25,32,33] that are known from their superb performance [2,34,35].
(b) iteratively updating the algorithm's parameters, using a sliding window, which enables online detection of anomalies.
(c) requiring only a fixed number of data points, which ensures a linear time complexity and a constant space complexity.

The rest of this paper is organised in the following format: Section 2 provides a preliminary background of autoencoders and LSTM networks. Section 3 contains the main contributions of this work and gives our unsupervised algorithm developed for anomaly detection in time-series. Section 4 describes the industrial case studies and gives a comparison of our algorithm with some of the state-of-the-art algorithms, and Section 5 concludes the paper.

2. Basic preliminaries

2.1. Deep autoencoders

Autoencoders are one of the most desirable types of artificial neural networks for unsupervised learning, trained to reconstruct the input at the output through back-propagation. They were originally developed for feature extraction and internal representation learning [36] and were later extended for anomaly detection [37] and non-linear dimensionality reduction [38]. Starting from some d-dimensional input $X \in \mathbb{R}^d$ that is drawn from an unknown distribution $q'(X)$, autoencoders are trained to learn a latent representation of the input, $Y \in \mathbb{R}^{d'}$, through either a stochastic or deterministic mapping $q'(Y|X;\theta)$ parameterised by a vector of parameters θ.

2.1.1. Maintaining the input information

Let p and q, depending on the context, denote both probability density functions or probability mass functions. Additionally, let $p(X)$, $p(Y)$, $p(X,Y)$, and $p(X|Y)$ denote the marginal probabilities, the joint, and the conditional probability for random variables X and Y. Moreover, following the literature convention, $p(x)$, $p(X|y)$, and $p(x|y)$ denote $p(X=x)$, $p(X|Y=y)$, and $p(X=x|Y=y)$.

Consider a deterministic mapping, $q(Y|X;\theta)$, from the input to the latent representation parameterised by θ. It is imperative that this representation retains sufficient information so that it can be used to reconstruct the input with minimal error. This criterion can be expressed as maximising the mutual information between the input and the latent representation, $\mathbb{I}(X;Y)$. Mathematically, $\mathbb{I}(X;Y)$ can be written in terms of marginal and conditional entropies:

$$\mathbb{I}(X;Y) = \mathrm{H}(X) - \mathrm{H}(X|Y) \ .$$

In the case where $Y = X$, then $\mathbb{I}(X;Y) = \mathrm{H}(X) - \mathrm{H}(X|X) = \mathrm{H}(X)$. If X and Y are independent ($X \perp\!\!\!\perp Y$), then knowing Y will not reveal any information about X and therefore, $\mathbb{I}(X;Y) = \mathrm{H}(X) - \mathrm{H}(X|Y) = \mathrm{H}(X) - \mathrm{H}(X) = 0$.

$$\underset{\theta}{\operatorname{argmax}}\ \mathbb{I}(X;Y) = \underset{\theta}{\operatorname{argmax}}\ \Big(\mathrm{H}(X) - \mathrm{H}(X|Y)\ .\Big)$$

It is easy to see that θ has no influence on $\mathrm{H}(X)$ and therefore the problem reduces to finding θ such that:

$$\underset{\theta}{\operatorname{argmax}}\ \mathbb{I}(X;Y) = \underset{\theta}{\operatorname{argmax}}\ (-\mathrm{H}(X|Y))\ . \qquad (1)$$

Theorem 1 (*Gibb's Inequality*). *For two probability distributions p and q, the following inequality for cross-entropy always holds:*

$$\mathrm{H}(p,q) \geq \mathrm{H}(p)\ .$$

The proof follows from the fact that the cross-entropy of distributions p and q can be defined as $\mathrm{H}(p,q) = H(p) + \mathbf{D}_{KL}(p \parallel q)$. Since the Kullback–Leibler divergence, $\mathbf{D}_{KL}(p \parallel q)$, is non-negative [39], it follows that $\mathrm{H}(p,q) \geq \mathrm{H}(p)$.

Therefore, following Theorem 1, a lower bound can be imposed on (1) using a conditional distribution $p(X|Y;\theta')$ parameterised by some θ'. Then the problem of maximising the mutual information can be reformulated as maximising the lower bound and since:

$$\mathbb{E}_{q(X,Y)}[\log q(X|Y)] = -H(X|Y)\ ,$$

where $\mathbb{E}$ denotes the Expectation, then:

$$\mathbb{E}_{q(X,Y)}[\log p(X|Y)] \leq \mathbb{E}_{q(X,Y)}[\log q(X|Y)]\ ,$$

which corresponds to maximising the mutual information. Moreover, assuming a deterministic mapping, $Y = g_\theta(X)$, the optimisation problem can be written as:

$$\max_{\theta,\theta'}\ \mathbb{E}_{q(X)}[\log p(X|g_\theta(X);\theta')]\ .$$

However, since $q(X)$ is unknown, an empirical average, $\mathbb{E}_{\tilde{q}(X)}$, over available samples can be used:

$$\max_{\theta,\theta'}\ \mathbb{E}_{\tilde{q}(X)}[\log p(X|g_\theta(X);\theta')]\ ,$$

which is the reconstruction error criterion for training autoencoders [40]. An autoencoder in its simplest form is composed of an *encoder* and a *decoder*. The encoder is the affine transformation $g_\theta : \mathbb{R}^d \to \mathbb{R}^{d'}$ that maps the input $\mathbf{x}$ to a latent representation $\mathbf{y}$, followed by a non-linearity:

$$g_\theta(\mathbf{x}) = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})\ ,$$

where $\sigma(x) := \frac{1}{1+e^{-x}}$ is the sigmoid function, $\mathbf{W} \in \mathbb{R}^{d' \times d}$ is the weight matrix and $\mathbf{b} \in \mathbb{R}^{d'}$ is the bias vector.

Then, the decoder is the transformation $f_{\theta'} : \mathbb{R}^{d'} \to \mathbb{R}^d$ that maps the latent representation $\mathbf{y}$ to $\mathbf{z} \in \mathbb{R}^d$, which is a reconstruction of $\mathbf{x}$:

$$f_{\theta'}(\mathbf{y}) = \sigma(\mathbf{W}'\mathbf{y} + \mathbf{b}')\ ,$$

where $\theta' = \{\mathbf{W}', \mathbf{b}'\}$ with appropriate dimensions. $\mathbf{z}$ should not be considered the exact replica of $\mathbf{x}$. However, minimising the reconstruction error $L(\mathbf{x},\mathbf{z}) \propto -\log p(\mathbf{x}|\mathbf{z})$ results in obtaining a θ' (if exists) that can reconstruct $\mathbf{x}$ with a high probability. The optimisation problem can be equivalently described with:

$$\underset{\theta,\theta'}{\operatorname{argmax}}\ \mathbb{E}_{\tilde{q}(X)}[\log p(X|g_\theta(X);\theta')]\ .$$

Several variations of autoencoders have been developed to learn richer representations and avoid learning an identity function. A trivial solution to ensure learning richer representations is by using deep autoencoders [19,40]. Deep autoencoders implement high degrees of non-linearity that enable them to learn complex functions compactly. Therefore, deep autoencoders can be combined with recurrent neural networks such as LSTMs to process sequence data.

2.2. Long short-term memory networks

The application of Deep Neural Network (DNN) architectures have resulted in significant achievements in many areas, especially in time-series modelling [41]. This success is mainly attributed to these networks' stacked architecture that allows a complex task to be partially solved in each layer. One variation of DNNs is the Recurrent Neural Networks (RNNs) when unfolded in time. The primary function of the layers in RNNs is to offer some memory rather than a hierarchical processing setting, which is seen in deep neural networks.

An important class of RNNs are LSTM networks that are suitable to represent sequential data. They address the vanishing gradient problem, seen in vanilla RNNs, through multiplicative gated units. LSTM networks have been applied widely in areas such as time-series analysis [42], natural language processing [43], and speech processing [44]. As in dense DNNs, a temporal hierarchy of the sequential data is best preserved when hidden layers are stacked to build deeper recurrent networks [45]. Moreover, stacked LSTM networks can be organised to form an autoencoder that can perform anomaly detection once trained on somewhat "clean" data.

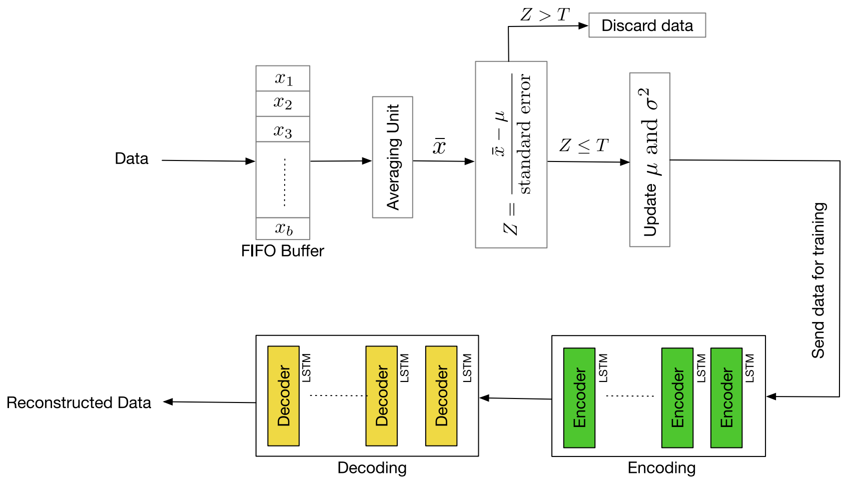

Fig. 1. Schematic of the ELSTMAE algorithm.

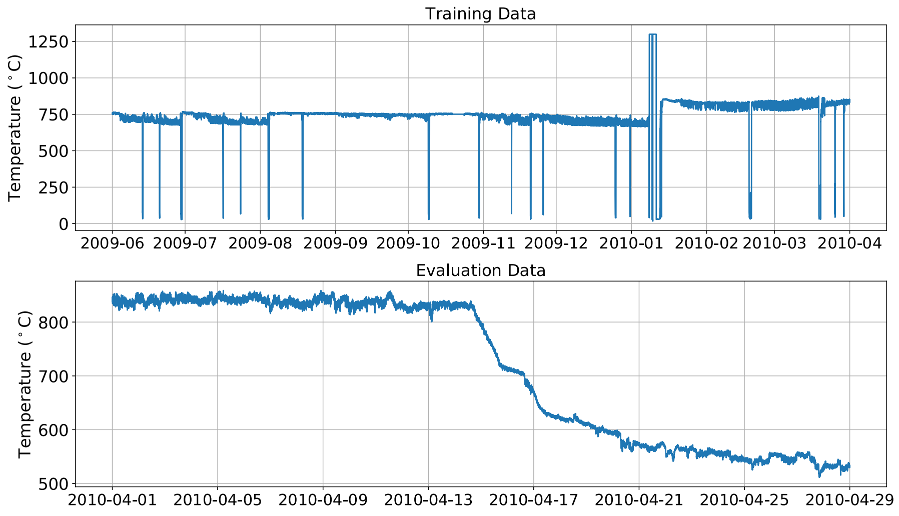

Fig. 2. Training and evaluation data obtained from an IGT burner-tip thermocouple.

3. Unsupervised labelling of the data

Through many observations of data, autoencoders learn to minimise reconstruction error between the input and output. When training is complete, any similar data fed to autoencoders produces a reasonably small reconstruction error. However, if the new data is statistically different from what is seen during the training process, autoencoders fail to reconstruct it properly at the output, resulting in a large error. It is this residual error that indicates the presence of an anomaly.

Autoencoders are typically trained on large historical datasets. In many industries, it is costly or impossible to have the training data labelled. If the training data contains anomalies, autoencoders learn to reconstruct them with a minimal error resulting in the dismissal of similar types when unseen (new) data is processed. Therefore, here, a probabilistic approach is introduced that assigns a probability to data points that indicates whether or not they are anomalous. More specifically, assume that all non-anomalous data points are drawn randomly and independently from an unknown probability distribution $\mathcal{U}$. It is intended to find the probability that whether a data point, x, being considered for training, is coming from $\mathcal{U}$. If $p(x \in \mathcal{U})$ satisfies some confidence level, it is passed for training, otherwise discarded. Hence, the model (an LSTM autoencoder) is trained only on non-anomalous data, and the reconstruction error is increased significantly if an anomaly is detected. This problem is termed as the *outlier-rejection problem*, and the main implementation challenge is the limited knowledge about $\mathcal{U}$, except the random samples drawn independently (with replacement). Although $\mathcal{U}$ cannot be approximated directly, the central limit theorem suggests these random samples can be used to infer whether or not the data is anomalous. This is by introducing a probability criterion that evaluates the likelihood that a data point is drawn from $\mathcal{U}$ —thereby addressing the outlier-rejection problem.

Definition 1. A sequence of cumulative distribution functions $\{F_n\}$ is said to converge in distribution to the CDF F, denoted by $F_n \Longrightarrow F$, if

$$\lim_{n\to\infty} F_n(x) = F(x) ,$$

for every continuity point x of F.

For completeness, the central limit theorem is given in the following theorem:

Theorem 2 (*Central Limit Theorem [46]*). *Let $\{X_1, \ldots, X_n\}$ be a sequence of independent and identically distributed random variables sampled from a distribution with mean μ and variance $\sigma^2 < \infty$. Denote the sample average by S_n, then as $n \to \infty$, the sample averages converges in distribution to $\mathcal{N}(\mu, \frac{\sigma}{\sqrt{n}})$.*

Implicitly, Theorem 2 states when repeated independent random samples are drawn from an unknown distribution with mean μ and variance σ^2, provided a large sampling size (conventionally $n \geq 30$, see e.g., [47]), the sampling distribution of the sample means approaches the normal distribution $\mathcal{N}(\mu, \frac{\sigma^2}{n})$. This idea paves the way for designing our outlier rejection algorithm. Using Theorem 2, the outlier rejection problem can be reformulated to assert, by means of a z-score, whether a data-point disturbs the underlying normal distribution of the collection of averages (termed as *population*) of sufficiently large samples drawn from $\mathcal{U}$.

3.1. The training procedure

A fixed-length First-In-First-Out (FIFO) buffer of length b captures the streaming data. The mean of the data points held inside, denoted by $\bar{x}$, is calculated upon arrival of each new data point (lines 1–7 of Algorithm 1). The collection of these means (termed as samples) form what is referred to as the "population". The hypothesis is that the data points stored in the buffer construct the "samples" drawn from $\mathcal{U}$. Since data points can appear repeatedly, the sampling process is considered to be *with replacement* which guarantees the samples' independence. As the population size gets larger (i.e., more data is received), Theorem 2 guarantees that the underlying distribution of the population (collection of means) converges to that of a normal unless the samples are drawn from another distribution (i.e., the underlying distribution of anomalies).

Therefore, the samples are compared, by means of a z-score, with the population at the arrival of each data-point (lines 8–18 of Algorithm 1). The resulting z-score is then thresholded (line 19 of Algorithm 1) for a desired confidence level [48] and the corresponding data-point is not passed for training if the threshold, T, is exceeded (lines 20–24 of Algorithm 1). Fig. 1 illustrates a diagram of this procedure.

For computational efficiency, we use a running algorithm to compute the z-score. The z-score is defined by

$$z := \frac{(\bar{x} - \mu)}{\text{standard error}} ,$$

where μ is the mean of the population and the standard error is $\sigma/\sqrt{b}$ and σ is the population standard deviation. The following updating procedure is used to compute the population's parameters on the fly

$$\mu_{new} := \mu_{old} + \frac{o - \mu_{old}}{n} , \qquad \sigma^2 := \frac{s}{n} ,$$

where μ_{new} is the population mean for the current time-step, μ_{old} is the population mean from the previous time-step, n is the length of the data received so far, o is the latest arrival in the buffer, and s is the auxiliary variable from the Welford's method [49] to compute the population's variance, and is calculated as

$$s := s + (o - \mu_{new}) \times (o - \mu_{old}) .$$

The time complexity of this procedure is constant in each iteration. For each data point, all operations are performed on a fixed number of points. Therefore, for n number of data points, the overall time complexity of the filtering algorithm is linear in the number of points (i.e., $\mathcal{O}(n)$). Furthermore, the memory space required throughout the entire filtering algorithm is constant since only the buffer, which has a fixed length, and a constant number of variables are required to be stored in memory. Therefore, the space complexity of this algorithm is constant, which is highly desirable. For further illustration, the pseudo-code of this procedure is presented in Algorithm 1.

Algorithm 1 Filtering Procedure

```
1:  # Initialise parameters
2:  $\mu = 0,\ \sigma = 0,\ s = 0,\ n = 0$
3:  while receiving data do
4:      $buffer \leftarrow$ new data-point
5:      if $len(buffer) = b$ then
6:        # Compute average of the buffer (sample)
7:          $\bar{x} \leftarrow mean(buffer)$
8:        # Get the first element out of the buffer
9:          $o \leftarrow buffer_{out}$
10:         $n \leftarrow n + 1$
11:       # Compute the estimated mean and variance of population
12:         $\mu_{old} \leftarrow \mu$
13:         $\mu_{new} \leftarrow \mu_{old} + (o - \mu_{old})/n$
14:         $s \leftarrow s + (o - \mu_{new}) \times (o - \mu_{old})$
15:         $\sigma \leftarrow \sqrt{\frac{s}{n}}$
16:         standard error = $\frac{\sigma}{\sqrt{b}}$
17:       # Compute the $z$-score
18:         $z \leftarrow (\bar{x} - \mu)/$standard error
19:         if $|z| < T$ then
20:             update $\mu$ and $\sigma^2$
21:             Pass data-point for training
22:         else
23:             $n \leftarrow n - 1$
24:             Reject data-point as outlier
```

4. Experimental results and benchmarking

Several real industrial scenarios are investigated to evaluate the performance of the proposed algorithm. The case studies presented in this paper are selected to reflect various types of anomalies that can appear (i.e., gradually-developing and abrupt). Moreover, we compare the performance of our algorithm, referred to as Enhanced LSTM Autoencoder (ELSTMAE), with four widely used algorithms (LSTMAD [23], LSTMED [25], VAE [32], and Luminol [33]) to demonstrate its efficacy. A virtual data streamer is developed to simulate the online streaming of the data.

In Algorithm 1, we set $T = 1.96$ to provide a 95% confidence in rejecting the abnormal data-points. This is because 95% of the area under a normal distribution is within 1.96 standard deviations of the mean. If other levels of confidence [48] are required, then the threshold can be configured accordingly.

4.1. Industrial gas turbine measurements

The first case study considers temperature measurements on an Industrial Gas Turbine (IGT) burner-tip thermocouple. Due to malfunctions, the measurements can drop or shoot up. Fig. 2 shows typical historical data (the training data) and gradually developing anomalous data (the evaluation data) obtained from one of the sensors.

The training data consists of anomalies that might indicate malfunctions, operational modes, or shut-downs. If these occurrences are frequent, as in this case, conventional autoencoders learn these anomalies as typical characteristics of the data and can reconstruct them at the output with no or a marginal reconstruction error. Therefore, these instances need to be processed during the training. Fig. 3 shows the probability density function plot for the training data. It is easy to see that the unidentifiable underlying distribution is not suitable for anomaly detection.

By applying Algorithm 1, therefore, the outliers are removed in the training process, and autoencoders learn the latent representation of non-anomalous data. It is now worth verifying how the

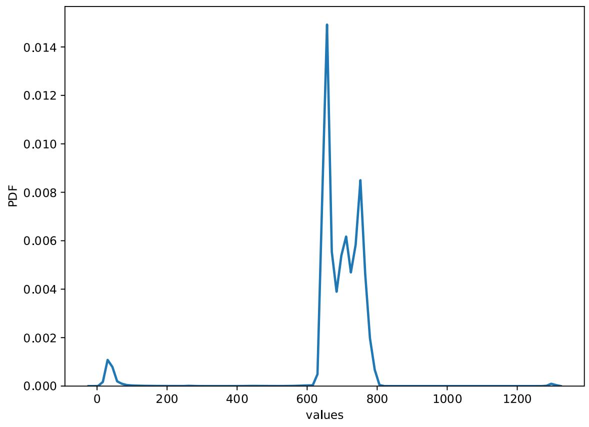

Fig. 3. PDF plot of the training data.

central limit theorem holds empirically (Fig. 4). As more samples are drawn from the data, the underlying distribution approaches to that of a normal (with known parameters), making it suitable for anomaly detection.

Upon completing training, the model is applied to assess the evaluation data, which has not been observed previously and exhibits some abnormalities. Fig. 5 shows a normalised figure of evaluation data and reconstruction errors generated by the four algorithms selected for benchmarking. The LSTMAD algorithm starts with a larger error than the ELSTMAE, but the error gradually approaches zero and gradually increases after the incident. If the error is thresholded, the alarm is raised several days after the incident. The LSTMED algorithm starts with a high error but drops to zero sharply around the incident and then gradually increases at a relatively slow pace, again missing the incident. The Luminol algorithm impressively produces almost no error when there are no anomalies. It also detects the anomaly with a slight delay by producing a large error. However, the error immediately goes to zero resulting in a large number of false negatives. This is because the algorithm learns the anomalies, so they are missed in the detection. The VAE algorithm starts with a comparatively higher but steady reconstruction error. After the incident, the error approaches zero and then gradually goes up (similar to the LSTMAD). The algorithm is also considerably late in detecting the anomaly. Contrary, our algorithm (ELSTMAE) generates a minimal error before the incident, and the error remains relatively steady as no abnormalities are observed. When the incident occurs, our algorithm is the only one that immediately generates a high error that increases at a fast pace — Thereby detecting the anomaly promptly. This is because by removing the outliers during the training, our algorithm ensures enough sensitivity to anomalous data to generate a high error when anomalies are observed.

4.2. CPU utilisation for an Amazon EC2 instance

The second case study considers the CPU utilisation for an Amazon EC2 instance (data from [50]). CPU surges often occur abruptly and can cause service outages. Therefore, the goal is set to detect future surges in CPU utilisation. Fig. 6 shows the training and evaluation data obtained for this case study.

The training data contains ten noticeable spikes, and it is desired to detect such cases should they show up later in the evaluation process. Training the model on the current training data results in similar problems discussed in Section 4.1. Fig. 7 shows the probability density function plot for the training data before Algorithm 1 is applied. It is easy to see that the unidentifiable underlying distribution is not suitable for anomaly detection.

Algorithm 1 is therefore applied to remove the outliers during the training. Fig. 8 shows how the central limit theorem holds empirically in this case as well. As more samples are drawn from the dataset, the underlying distribution approaches to that of a normal (with known parameters) making it suitable for anomaly detection.

Then, the processed data is used to train an LSTM autoencoder. Upon completing training, the model is used to assess the

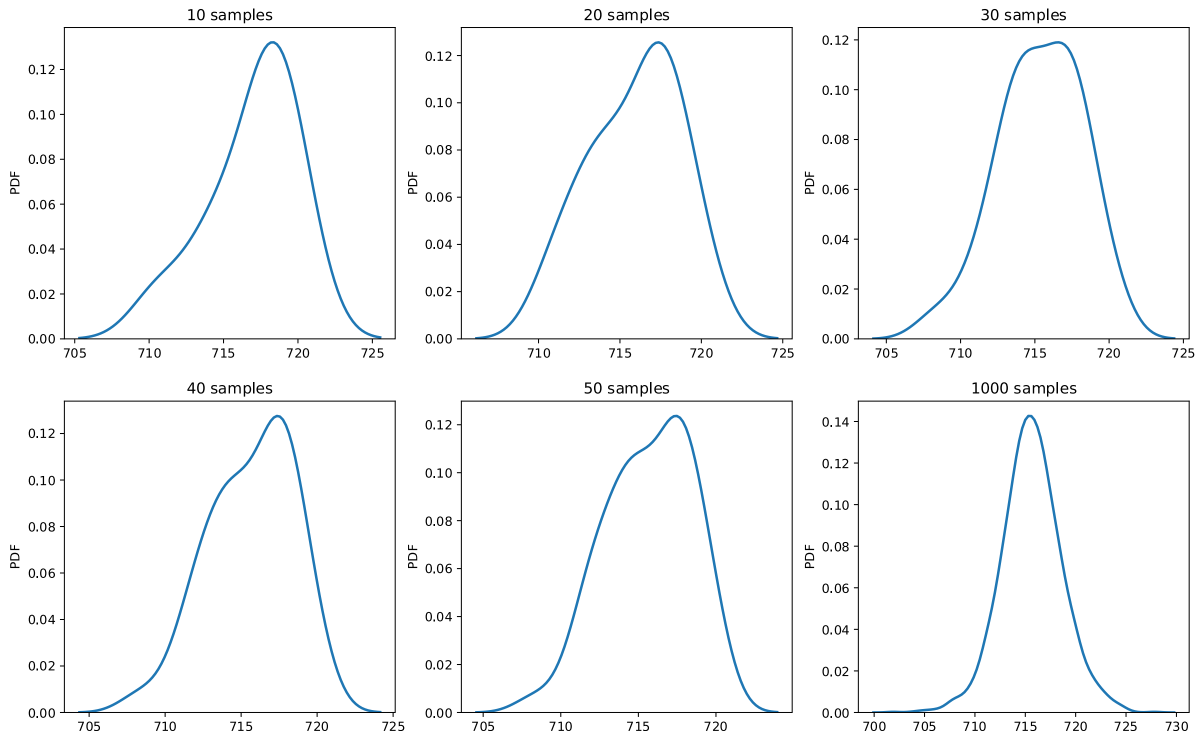

Fig. 4. PDF plots for various sampling times show as the number of samples increases, the underlying distribution approaches to normal.

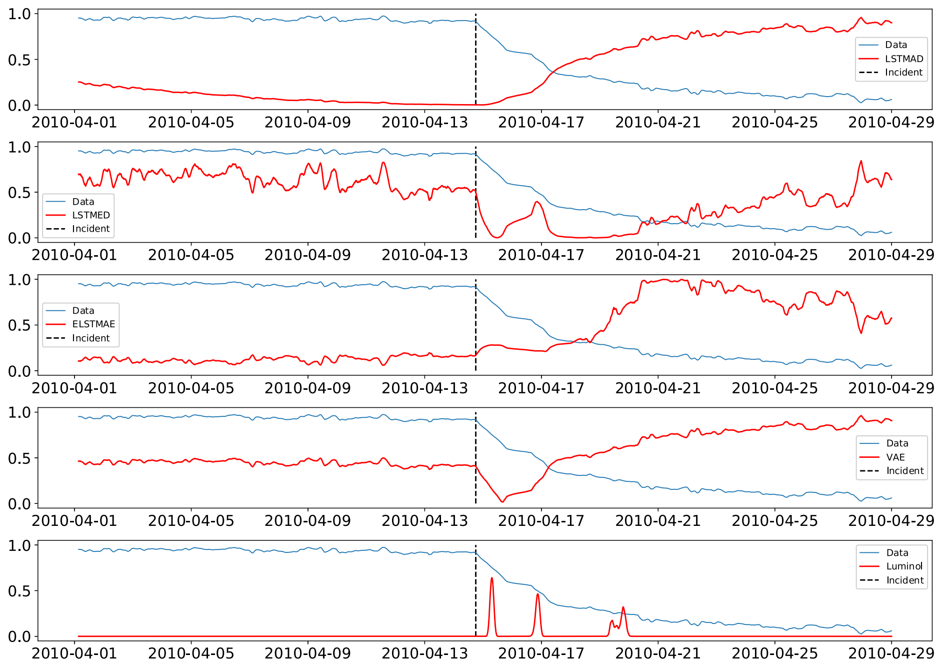

Fig. 5. Normalised model reconstruction error for evaluation data.

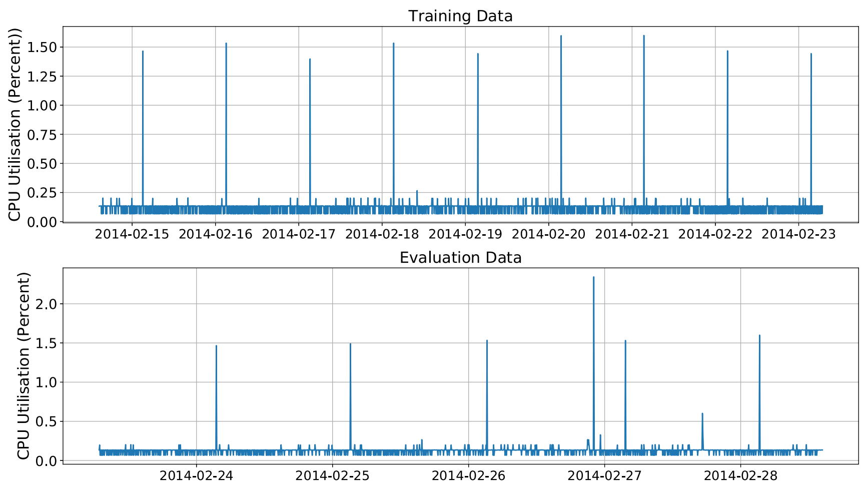

Fig. 6. Training and evaluation data for an Amazon EC2 instance CPU utilisation.

Table 1
Anomaly detection results on two datasets.

Method	AWS			Siemens		
	Pre	Rec	F_1	Pre	Rec	F_1
LSTMAD	0.07	0.70	0.13	0.94	0.83	0.91
LSTMED	0.11	0.11	0.10	0.50	0.99	0.67
Luminol	0.85	0.6	0.71	0.10	0.10	0.19
VAE	0.90	1.0	0.95	0.86	0.70	0.82
ELSTMAE	0.90	1.0	0.95	0.99	0.95	0.97

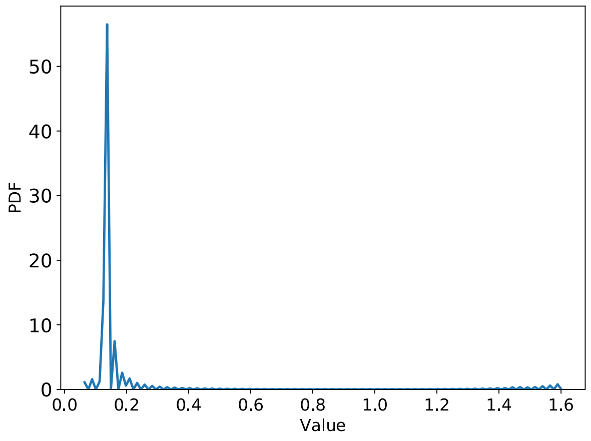

Fig. 7. PDF plot of the training data.

evaluation data, which is known to exhibit abnormalities. Fig. 9 shows the normalised reconstruction error generated by the four algorithms considered here. The evaluation data consists of seven spikes, one of which has a relatively smaller magnitude. While all algorithms generate a higher error when spikes occur, the error for small spikes is considerably small for the LSTMAD, and LSTMED, making it difficult to threshold. VAE, Luminol and

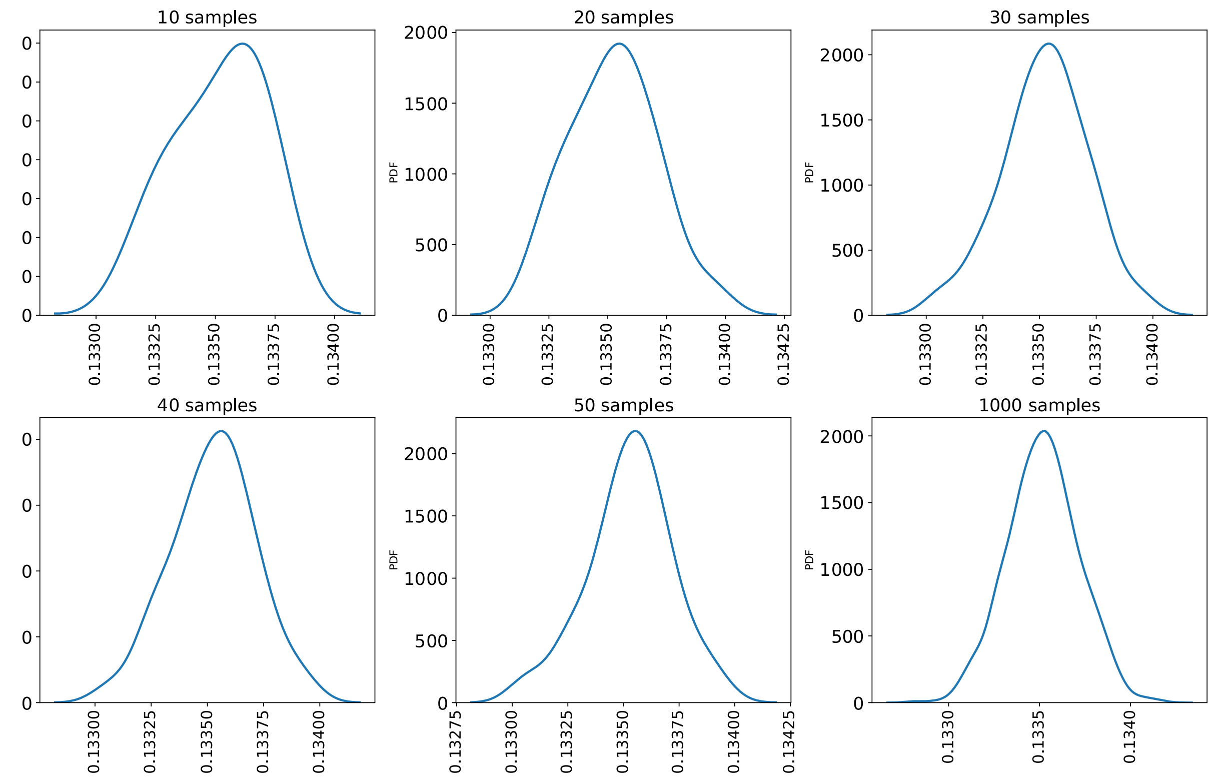

Fig. 8. PDF plots for various sampling times show as the number of samples increases, the underlying distribution approaches to normal.

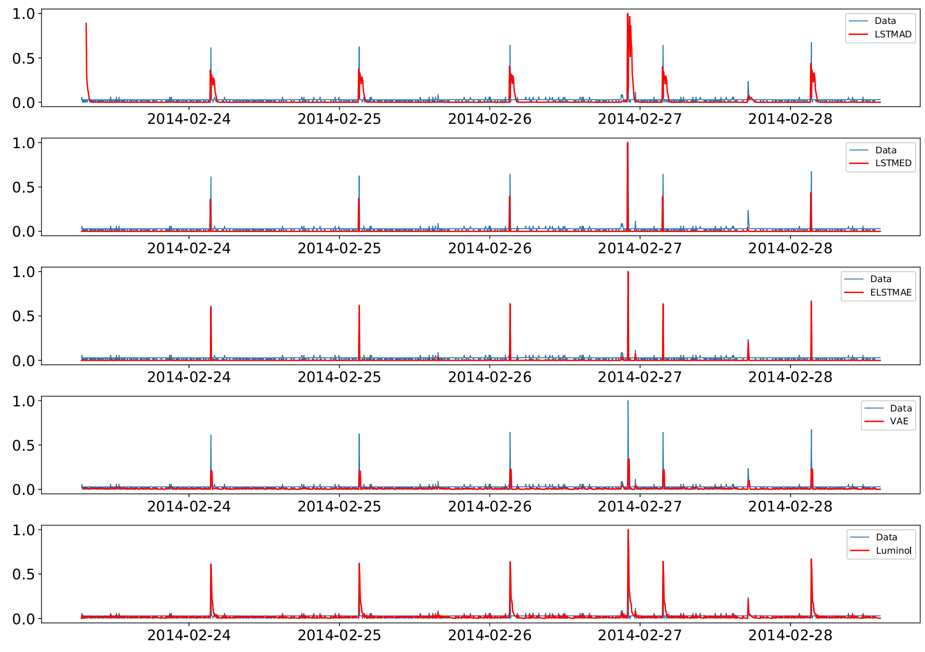

Fig. 9. Normalised model reconstruction error for evaluation data.

ELSTMAE perform similarly in detecting the anomalies. However, the Luminol algorithm that uses a sign test to detect anomalous data produces many false positives due to noise.

For further benchmarking, **F₁** score, **Precision**, and **Recall** are used as three metrics to evaluate the performance of our algorithm (ELSTMAE) in the experiments above.

$$Precision = \frac{T_p}{F_p}\ , \quad Recall = \frac{T_p}{T_p + F_n}\ , \quad F_1 = 2 \times \frac{Precision\ .\ Recall}{Precision + Recall}\ ,$$

where T_p, F_p, and F_n are the number of true positives, false positives, and false negatives, respectively.

Following the benchmarking convention in the literature (e.g., [51]), all the anomaly detection methods selected for benchmarking are applied on each dataset and the corresponding F_1-scores are reported in Table 1 for each method at the best threshold.

The comparison shows that the F_1 scores for LSTMAD, LSTMED, and Luminol are not stable across the datasets, and the results contain many false positives and false negatives. Meanwhile, the scores of VAE and ELSTMAE are much more stable. VAE and ELSTMAE perform very similarly with high accuracy on the dataset that consists of abrupt changes. The table also reveals that the LSTM-based methods perform better when dealing with gradually developing anomalies (Siemens dataset). The reason for the superior performance of the ELSTMAE in both datasets is attributed to the statistical data-filtering algorithm, where anomalies are removed from the training dataset — resulting in higher detection accuracy.

5. Conclusions

An online unsupervised algorithm for anomaly detection in time-series is developed. A probability criterion based on the classical central limit theorem is introduced that allows the labelling of the streaming data. Two real case studies were considered. In the first case, where a malfunction had gradually developed, the algorithm proved to detect the incident (Fig. 5). Similarly, in the second case, the algorithm proved to detect abrupt changes in the CPU utilisation data for an Amazon EC2 instance (Fig. 9).

The main advantages of our approach include (**I**) Swift detection of abrupt and gradually developing anomalies. (**II**) Unsupervised labelling of the training set, which is often costly to obtain. (**III**) The anomaly label is associated with a confidence level, which can be tuned to match the system's specific requirements.

The main limitations of this approach concern the weak assumptions regarding the statistical labelling procedure. I.e., availability and reliability of the historical data. It is assumed that the underlying system has been working for some time, and sufficient data has been observed. Additionally, it is also assumed that the underlying system is correctly designed so that an abnormal operation's probability is considerably lower. Moreover, LSTM architectures require a high memory for training. Therefore, coupling to external memory might be necessary to avoid any lags when processing high volume/velocity data [52].

For our future work, we are considering multivariate features for more robust anomaly detection. This is because integrating multiple data consisting of signatures of anomalies results in more consistent and accurate information than the case where only one feature is available. As a result, we are also working on architectures that are better suited for multidimensional data.

CRediT authorship contribution statement

Sepehr Maleki: Conceptualisation, Methodology, Investigation, Software, Validation, Formal analysis, Data curation, Writing - original draft. **Sasan Maleki:** Conceptualisation, Methodology, Investigation, Formal analysis, Software, Writing - review & editing. **Nicholas R. Jennings:** Methodology, Writing - review & editing.

Declaration of competing interest

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

Acknowledgements

The authors would like to thank Siemens Industrial Turbomachinery, Lincoln, U.K., for providing access to real-time data to support the research outcomes.

References

[1] H. Ren, B. Xu, Y. Wang, C. Yi, C. Huang, X. Kou, T. Xing, M. Yang, J. Tong, Q. Zhang, Time-series anomaly detection service at microsoft, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Association for Computing Machinery, New York, NY, USA, 2019, pp. 3009–3017.

[2] R. Chalapathy, S. Chawla, Deep learning for anomaly detection: A survey, 2019, arXiv:1901.03407.

[3] M. Munir, S.A. Siddiqui, A. Dengel, S. Ahmed, DeepAnT: A deep learning approach for unsupervised anomaly detection in time series, IEEE Access 7 (2019) 1991–2005.

[4] R. Chalapathy, A.K. Menon, S. Chawla, Anomaly detection using one-class neural networks, in: KDD 2018, London, United Kingdom, 2018.

[5] D. Xu, E. Ricci, Y. Yan, J. Song, N. Sebe, Learning deep representations of appearance and motion for anomalous event detection, in: Proceedings of the British Machine Vision Conference, BMVC, BMVA Press, 2015, pp. 8.1–8.12, http://dx.doi.org/10.5244/C.29.8.

[6] S. Maleki, P. Rapisarda, L. Ntogramatzidis, E. Rogers, Failure identification for 3D linear systems, Multidimens. Syst. Signal Process. 26 (2) (2015) 481–502.

[7] S. Maleki, P. Rapisarda, E. Rogers, Failure identification for linear repetitive processes, Multidimens. Syst. Signal Process. 26 (4) (2015) 1037–1059.

[8] S. Cho, J. Jiang, A fault detection and isolation technique using nonlinear support vectors dichotomizing multi-class parity space residuals, J. Process Control 82 (2019) 31–43.

[9] D.S. Pillai, N. Rajasekar, Metaheuristic algorithms for PV parameter identification: A comprehensive review with an application to threshold setting for fault detection in PV systems, Renew. Sustain. Energy Rev. 82 (2018) 3503–3525.

[10] Y. Sun, G.G. Yen, Z. Yi, Evolving unsupervised deep neural networks for learning meaningful representations, IEEE Trans. Evol. Comput. 23 (1) (2018) 89–103.

[11] D.R. Ly, A. Grossi, C. Fenouillet-Beranger, E. Nowak, D. Querlioz, E. Vianello, Role of synaptic variability in resistive memory-based spiking neural networks with unsupervised learning, J. Phys. D: Appl. Phys. 51 (44) (2018) 444002.

[12] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM Comput. Surv. 41 (3) (2009) 15:1–15:58.

[13] C. De Stefano, C. Sansone, M. Vento, To reject or not to reject: That is the question-an answer in case of neural classifiers, Trans. Syst. Man Cybern. C 30 (1) (2000) 84–94.

[14] S. Aminikhanghahi, D.J. Cook, A survey of methods for time series change point detection, Knowl. Inf. Syst. 51 (2) (2017) 339–367.

[15] R. Prescott Adams, D.J.C. MacKay, Bayesian online changepoint detection, 2007, arXiv e-prints, arXiv:0710.3742.

[16] S. Maleki, C. Bingham, Y. Zhang, Development and realization of changepoint analysis for the detection of emerging faults on industrial systems, IEEE Trans. Ind. Inf. 12 (2016) 1180–1187.

[17] E.J. Candès, X. Li, Y. Ma, J. Wright, Robust principal component analysis? J. ACM 58 (3) (2011) 11:1–11:37.

[18] Y. Zhang, L. Duan, M. Duan, A new feature extraction approach using improved symbolic aggregate approximation for machinery intelligent diagnosis, Measurement 133 (2019) 468–478.

[19] C. Zhou, R.C. Paffenroth, Anomaly detection with robust deep autoencoders, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '17, ACM, New York, NY, USA, 2017, pp. 665–674.

[20] Y. Zhao, B. Deng, C. Shen, Y. Liu, H. Lu, X.-S. Hua, Spatio-temporal autoencoder for video anomaly detection, in: Proceedings of the 25th ACM International Conference on Multimedia, 2017, pp. 1933–1941.

[21] Z. Chen, C.K. Yeo, B.S. Lee, C.T. Lau, Autoencoder-based network anomaly detection, in: 2018 Wireless Telecommunications Symposium, WTS, IEEE, 2018, pp. 1–5.

[22] Z. Che, S. Purushotham, K. Cho, et al., Recurrent neural networks for multivariate time series with missing values, Nature: Sci. Rep. (2018) 6085.

[23] P. Malhotra, L. Vig, G. Shroff, P. Agarwal, Long short term memory networks for anomaly detection in time series, in: Proceedings, Vol. 89, Presses universitaires de Louvain, 2015.

[24] H. Nguyen, K.P. Tran, S. Thomassey, M. Hamad, Forecasting and anomaly detection approaches using LSTM and LSTM Autoencoder techniques with the applications in supply chain management, Int. J. Inf. Manage. (2020) 102282.

[25] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, G. Shroff, LSTM-based encoder-decoder for multi-sensor anomaly detection, in: ICML 2016 Anomaly Detection Workshop, NY, USA, 2016.

[26] M. Said Elsayed, N.-A. Le-Khac, S. Dev, A.D. Jurcut, Network anomaly detection using LSTM based autoencoder, in: Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks, 2020, pp. 37–45.

[27] M. Wang, Z. Wang, J. Lu, J. Lin, Z. Wang, E-lstm: An efficient hardware architecture for long short-term memory, IEEE J. Emerg. Sel. Top. Circuits Syst. 9 (2) (2019) 280–291.

[28] S. Wang, Z. Li, C. Ding, B. Yuan, Q. Qiu, Y. Wang, Y. Liang, C-LSTM: Enabling efficient LSTM using structured compression techniques on FPGAs, in: Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2018, pp. 11–20.

[29] J. Han, H. Liu, M. Wang, Z. Li, Y. Zhang, ERA-LSTM: An efficient ReRAM-based architecture for long short-term memory, IEEE Trans. Parallel Distrib. Syst. 31 (6) (2019) 1328–1342.

[30] S. Cao, C. Zhang, Z. Yao, W. Xiao, L. Nie, D. Zhan, Y. Liu, M. Wu, L. Zhang, Efficient and effective sparse LSTM on FPGA with bank-balanced sparsity, in: Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2019, pp. 63–72.

[31] M. Wielgosz, A. Skoczeń, M. Mertik, Using LSTM recurrent neural networks for monitoring the LHC superconducting magnets, Nucl. Instrum. Methods Phys. Res. A 867 (2017) 40–50.

[32] C.P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, A. Lerchner, Understanding disentangling in β-VAE, in: Neural Information Processing Systems (NIPS) Workshop on Learning Disentangled Representations: From Perception to Control, 2017.

[33] R. Maheshwari, Y. Yang, R. Hou, B. Li, L. Zhang, Luminol: Anomaly detection and correlation library, URL https://github.com/linkedin/luminol.

[34] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, T. Soderstrom, Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Association for Computing Machinery, New York, NY, USA, 2018, pp. 387–395.

[35] I. Khemakhem, D. Kingma, R. Monti, A. Hyvarinen, Variational autoencoders and nonlinear ICA: A unifying framework, in: S. Chiappa, R. Calandra (Eds.), Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, in: Proceedings of Machine Learning Research, vol. 108, PMLR, 2020, pp. 2207–2217.

[36] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representations by error propagation, in: Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations, MIT Press, Cambridge, MA, USA, 1986, pp. 318–362.

[37] N. Japkowicz, C. Myers, M. Gluck, A novelty detection approach to classification, in: Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 1, IJCAI'95, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995, pp. 518–523.

[38] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.

[39] S. Kullback, Information Theory and Statistics, in: Dover Books on Mathematics, Dover Publications, 1997.

[40] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion, J. Mach. Learn. Res. 11 (2010) 3371–3408.

[41] M. Längkvist, L. Karlsson, A. Loutfi, A review of unsupervised feature learning and deep learning for time-series modeling, Pattern Recognit. Lett. 42 (2014) 11–24.

[42] F. Karim, S. Majumdar, H. Darabi, S. Chen, LSTM fully convolutional networks for time series classification, IEEE Access 6 (2017) 1662–1669.

[43] T.-H. Wen, M. Gasic, N. Mrkšić, P.-H. Su, D. Vandyke, S. Young, Semantically conditioned LSTM-based natural language generation for spoken dialogue systems, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Lisbon, Portugal, 2015, pp. 1711–1721.

[44] F. Weninger, H. Erdogan, S. Watanabe, E. Vincent, J. Le Roux, J.R. Hershey, B. Schuller, Speech enhancement with LSTM recurrent neural networks and its application to noise-robust ASR, in: International Conference on Latent Variable Analysis and Signal Separation, Springer, 2015, pp. 91–99.

[45] M. Hermans, B. Schrauwen, Training and analysing deep recurrent neural networks, in: Advances in Neural Information Processing Systems, 2013, pp. 190–198.

[46] Central limit theorem, in: The Concise Encyclopedia of Statistics, Springer New York, New York, NY, 2008, pp. 66–68, http://dx.doi.org/10.1007/978-0-387-32833-1_50.

[47] R.V. Hogg, Instructor's solutions manual probability and statistical inference, 2016.

[48] R. Sprinthall, Basic Statistical Analysis, Allyn and Bacon, 2003.

[49] B.P. Welford, Note on a method for calculating corrected sums of squares and products, Technometrics 4 (3) (1962) 419–420.

[50] S. Ahmad, A. Lavin, S. Purdy, Z. Agha, The Numenta Anomaly Benchmark (NAB), URL https://github.com/numenta/NAB.

[51] S. Lin, R. Clark, R. Birke, S. Schonborn, N. Trigoni, S. Roberts, Anomaly detection for time series using VAE-LSTM hybrid model, in: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, 2020, pp. 4322–4326, http://dx.doi.org/10.1109/ICASSP40776.2020.9053558.

[52] A. Graves, G. Wayne, I. Danihelka, Neural Turing machines, 2014, arXiv preprint arXiv:1410.5401.