


Article

Semi-Supervised Time Series Anomaly Detection Based on Statistics and Deep Learning

Jehn-Ruey Jiang , Jian-Bin Kao and Yu-Lin Li

Department of Computer Science and Information Engineering, National Central University, Taoyuan City 32001, Taiwan; 106522115@cc.ncu.edu.tw (J.-B.K.); 107522120@cc.ncu.edu.tw (Y.-L.L.)

* Correspondence: jrjiang@csie.ncu.edu.tw

Abstract: Thanks to the advance of novel technologies, such as sensors and Internet of Things (IoT) technologies, big amounts of data are continuously gathered over time, resulting in a variety of time series. A semi-supervised anomaly detection framework, called Tri-CAD, for univariate time series is proposed in this paper. Based on the Pearson product-moment correlation coefficient and Dickey–Fuller test, time series are first categorized into three classes: (i) periodic, (ii) stationary, and (iii) non-periodic and non-stationary time series. Afterwards, different mechanisms using statistics, wavelet transform, and deep learning autoencoder concepts are applied to different classes of time series for detecting anomalies. The performance of the proposed Tri-CAD framework is evaluated by experiments using three Numenta anomaly benchmark (NAB) datasets. The performance of Tri-CAD is compared with those of related methods, such as STL, SARIMA, LSTM, LSTM with STL, and ADSaS. The comparison results show that Tri-CAD outperforms the others in terms of the precision, recall, and F₁-score.

Keywords: anomaly detection; autoencoder; deep learning; Internet of Things; sensors; time series; wavelet transform



Citation: Jiang, J.-R.; Kao, J.-B.; Li, Y.-L. Semi-Supervised Time Series Anomaly Detection Based on Statistics and Deep Learning. *Appl. Sci.* **2021**, *11*, 6698. <https://doi.org/10.3390/app11156698>

Academic Editor: Markus Goldstein

Received: 5 June 2021

Accepted: 18 July 2021

Published: 21 July 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Anomaly detection involves identifying data that do not conform to the pattern as expected [1,2]. It is also known as outlier detection [3,4] or novelty detection [5]. The research of anomaly detection can be traced back to the 18th century when Bernoulli commented on the widespread practice of discarding discordant data in the absence of a priori information in 1777 [6,7]. Undoubtedly, anomaly detection has attracted much research attention, bring about plenty of research results that can be utilized in various applications. Typical applications of anomaly detection research include outlier detection for big data [8], Internet of Things (IoT) [9], wireless sensor networks (WSNs) [10], network anomaly/intrusion/attack detection [11–16], anomaly detection for manufacturing [17–21], video surveillance anomaly detection [22,23], road network/highway traffic anomaly detection [24,25], credit card fraud identification [26,27], insurance fraud detection [28], ECG (electrocardiogram) monitoring [29], suspicious financial transaction detection [30], social media rumor detection [31], and so on.

Thanks to the advances of novel technologies, such as sensor technologies and information and communication technologies, big amounts of data are continuously gathered over time. Data that are recorded chronologically constitute a time series. Typical examples of time series include the following gathered data: hourly temperatures of a machine, daily stock prices of a company, weekly interest rates, monthly sales of a store, and yearly GDP (gross domestic product) of a nation. Time series anomaly detection is very important; it is one of fundamental time series analysis tasks. Many studies have focused on such a research field in the last decades. The readers are referred to the survey papers [32–35] for details.

Time series can be categorized into two types: univariate time series and multivariate time series. The former can be regarded as a sequence of scalar values, whereas the latter as a sequence of multi-dimensional vectors. This paper focuses on univariate time series and proposes a semi-supervised anomaly detection framework, called Tri-CAD (standing for tri-class anomaly detection), for such time series. Tri-CAD is semi-supervised, as explained below.

Anomaly detection methods can be broadly categorized into three types: supervised, unsupervised, and semi-supervised methods. A supervised method takes datasets that have data labeled as normal or abnormal (anomalous) to train or model a classifier. An unsupervised method takes unlabeled data as input under the assumption that the majority of data are normal. Data are identified as abnormal if they fit least to the rest of data. A semi-supervised method takes datasets of normal data as input to train a model representing the normal behavior of data. Test data are then fed into the trained model to check if they deviate from the normal behavior of data. If so, they are regarded as abnormal. Tri-CAD is semi-supervised in the sense that it takes the first D (say, $D = 1000$) data of time series as input under the assumption that the first D data are normal. Tri-CAD adopts this assumption because of the following practical considerations. Time series associated with well-functioning systems usually contain a large amount of normal data. Abnormal data are very rare; they are even never observed or kept in time series. Specifically, Tri-CAD takes the first D data as training data (also called normal data), and the rest of the data are taken as test data.

By the first D data, Tri-CAD divides time series into three classes according to the Pearson product-moment correlation coefficient [36] and Dickey–Fuller test [37]. The three classes of time series are (Class 1) periodic time series, (Class 2) stationary time series, and (Class 3) non-periodic and non-stationary time series. Afterwards, different approaches using statistics and deep learning mechanisms are applied to different classes of data to detect anomalies. Tri-CAD further divides first D data into overlapping sliding windows for calculating specific statistic characteristics, which in turn are used for calculating the mean μ and the standard deviation σ of all sliding windows of data. The same sliding window is applied to the test data for calculating the same statistic characteristics. A test datum is assumed to be anomalous if its associated statistic characteristic c of the sliding window deviates from the mean μ over a certain factor λ of the standard deviation σ (i.e., $|c - \mu| > \lambda\sigma$). Note that different λ value corresponds to different occurrence probability of such an anomalous datum. For example, $\lambda = 3$ corresponds to 0.2699796% of occurrence, $\lambda = 3.890592$ corresponds to 0.01% of occurrence, $\lambda = 4.417173$ corresponds to 0.001% of occurrence, and $\lambda = 6$ corresponds to 0.0000001973% of occurrence.

For dealing with periodic time series data, the moving averages of skews per period are calculated. The mean and standard deviation of the moving averages of the normal D data are derived. If the moving average of a test datum deviates too much from the mean, an anomaly is assumed to occur. For tackling stationary time series data, the means of the data in two sliding time windows are calculated. One window is larger and called the global window; the other is smaller and called the local window. The ratio of the two means is subsequently calculated for each pair of windows. Similarly, an anomaly is assumed to happen if the ratio corresponding to a test datum deviates too much from the ratio mean of normal data. For non-periodic and non-stationary time series, the data are first processed with the discrete wavelet transform (DWT) [38] to be time–frequency coefficients. The coefficients are in turn used to train a deep learning autoencoder (AE) model [39] that can reconstruct the coefficients with low reconstruction errors. Likewise, if a reconstruction error deviates from the error mean significantly, then it is assumed that an anomaly occurs.

This research applies three Numenta anomaly benchmark (NAB) [40] real-world datasets to evaluate the performance of Tri-CAD in terms of the precision, recall, and F_1 -score. It also compares Tri-CAD with related methods, such as the SARIMA [41], STL [42],

LSTM [43], LSTM with STL [44], and ADSaS [44] methods. The comparisons show that Tri-CAD outperforms the others in all three datasets.

The contribution of this research is threefold. First, it proposes the Tri-CAD framework to perform anomaly detection by first classifying time series into different classes, and then applying different anomaly detection methods that are suitable for different classes of data. Second, it implements the Tri-CAD framework and applies the implementation to real-world datasets to validate the concept of the framework. Third, this research conducts performance evaluation to show that Tri-CAD has the best performance when compared with other related methods.

Note that this paper is an extended version of two research papers [45,46]. The Tri-CAD framework is similar to that proposed in [45]. However, it will be shown later that Tri-CAD has a significant difference from the framework proposed in [45]. Moreover, Tri-CAD also integrates an anomaly detection mechanism called wavelet autoencoder anomaly detection (WAAD), proposed in [46]. Nevertheless, it will be shown later that Tri-CAD adopts a different threshold setting strategy when employing WAAD.

The rest of this paper is organized as follows. Section 2 introduces some background knowledge. The proposed univariate time series anomaly detection framework is elaborated on in Section 3. The performance of the proposed Tri-CAD framework is evaluated and compared with those of related methods in Section 4. Section 5 finally concludes the paper.

2. Background Knowledge

2.1. Time Series and Related Models

A time series $X = (x_1, x_2, \dots)$ is a sequence of observations (or data) ordered chronologically, where x_t is recorded at a specific time point t , where t is a positive integer [47]. It is usually assumed that time points are equally separated. Based on the properties of observations, we have two types of time series. On the one hand, if observations of a time series are scalar values, then the time series is a univariate time series. On the other hand, if observations are multi-dimensional vectors, then the time series is a multivariate time series. This paper focuses on univariate time series; therefore, we use “time series” to stand for “univariate time series” in the following context.

Time series can be either stationary or non-stationary. According to [47], the definition of a stationary time series is given below. A time series X is strictly stationary if the probability distribution of X does not change with time shifts. In other words, for every m and n , the distributions of $Y = (x_1, \dots, x_n)$ and $Z = (x_{1+m}, \dots, x_{n+m})$ are the same, where Y and Z are subsequence of X . A time series X is weakly stationary if its mean $E(\cdot)$, variance $Var(\cdot)$, and covariance $Cov(\cdot)$ do not change by time shifts. In other words, for every m and n , $E(Y) = E(Z) = \mu$, $Var(Y) = Var(Z) = \sigma^2$, and $Cov(Y, Z) = \gamma(|m - n|)$ for some function $\gamma(\cdot)$. In this paper, a time series is assumed to be stationary if it is weakly stationary. The next subsection will show how to use the Dickey–Fuller test to check the stationarity of a time series.

Stationary time series can be easily formulated by the auto-regressive (AR) [48], moving average (MA) [48], and auto-regressive and moving average (ARMA) [48] models, whereas non-stationary time series can be properly formulated or processed by the auto-regressive integrated moving average (ARIMA) [48] model and the seasonal auto-regressive integrated moving average (SARIMA) [41] model, and the seasonal-trend decomposition based on LOESS (STL) method [42]. Below, the models and the method are described one by one.

Let $X = (x_1, x_2, \dots)$ be a stationary time series and x_t in X be an observation recorded at a specific time point t , where t is a positive integer. In consideration of p earlier lags $x_{t-1}, x_{t-2}, \dots, x_{t-p}$ and the white noise (or random error) e_t at time t with the zero mean and a bounded variance, x_t can be formulated according to the auto-regressive AR(p) model, as shown in (1).

$$x_t = a_1 x_{t-1} + a_2 x_{t-2} + \dots + a_p x_{t-p} + e_t, \quad (1)$$

where p is the order and a_1, a_2, \dots, a_p are coefficients or parameters of the AR model.

In consideration of the random error e_t at time t and q pervious random errors $e_{t-1}, e_{t-2}, \dots, e_{t-q}$, the deviation of x_t from the mean μ of the whole series can be formulated according to the moving-average MA(q) model, as shown in (2).

$$x_t - \mu = e_t + b_1 e_{t-1} + b_2 e_{t-2} + \dots + b_q e_{t-q}, \quad (2)$$

where q is the order and b_1, b_2, \dots, b_q are coefficients or parameters of the MA model.

In consideration of p earlier lags $x_{t-1}, x_{t-2}, \dots, x_{t-p}$, q pervious random errors $e_{t-1}, e_{t-2}, \dots, e_{t-q}$, and the random error e_t , x_t can be formulated according to the auto-regressive and moving-average ARMA(p, q) model, as shown in (3).

$$x_t - (a_1 x_{t-1} + a_2 x_{t-2} + \dots + a_p x_{t-p}) = e_t + b_1 e_{t-1} + b_2 e_{t-2} + \dots + b_q e_{t-q}, \quad (3)$$

where p is the order and a_1, a_2, \dots, a_p are coefficients of the AR model, and q is the order and b_1, b_2, \dots, b_q are coefficients of the MA model. By introducing the lag operator L , where $Lx_t = x_{t-1}$, and $L(Lx_t) = L^2 x_t = x_{t-2}, \dots$, and so on, we can rewrite ARMA(p, q) according to (4).

$$(1 - \sum_{i=1}^p a_i L^i) x_t = (1 + \sum_{i=1}^q b_i L^i) e_t \quad (4)$$

Let $X = (x_1, x_2, \dots)$ be a non-stationary time series and x_t in X be an observation recorded at a specific time point t , where t is a positive integer. As X is not stationary, it cannot be formulated by AR, MA, or ARMA models. However, Box and Jenkins [49] suggested that differencing a non-stationary time series one or more times can make it stationary. To be specific, by differencing the non-stationary time series X for d times, X can still be properly formulated by the ARIMA(p, d, q) model, as described below. Note that $(1 - L)x_t = x_t - x_{t-1}$ is first-order difference, and $(1 - L)^d x_t = d^{th}$ -order difference. We have the following Formula (5).

$$(1 - \sum_{i=1}^p a_i L^i) (1 - L)^d x_t = (1 + \sum_{i=1}^q b_i L^i) e_t \quad (5)$$

The Box–Jenkins approach leads to the ARIMA model to formulate a non-stationary time series. It has three steps [48], as elaborated below.

Step 1: Determining the order d of differencing to make the time series stationary:

This step can be achieved by repeatedly differencing the time series and by applying the Dickey–Fuller test to check whether or not the time series has become stationary. The Dickey–Fuller test will be described later in the next subsection.

Step 2: Identifying the model:

This step can be achieved by applying the auto correlation function (ACF) and the partial auto correlation function (PACF) to determine which model is most suitable to formulate the time series. Possible models are AR(p, d), MA(d, q), and ARMA(p, d, q).

Step 3: Estimating the parameters:

No matter which of AR(p, d), MA(d, q), and ARMA(p, d, q) is determined in the previous step, it is a linear regression model. Therefore, this step can be achieved by applying optimization mechanisms, such as the maximum likelihood estimator, to find model coefficients just like finding linear regression coefficients [50].

As just mentioned, the ARIMA model formulates the time series trend with three trend parameters: the auto-regression order p , the difference order d , and the moving average order q . By adding four seasonal parameters, the SARIMA (seasonal auto-regressive integrated moving average or seasonal ARIMA) model further considers the seasonal component by adding four seasonal parameters: the seasonal auto-regressive order P , the seasonal difference order D , the seasonal moving average order Q , and the number m of

time points for a single seasonal period. A SARIMA model is specified as $SARIMA(p, d, q)(P, D, Q)_m$. In addition to d -order differencing of ARIMA, seasonal ARIMA or SARIMA also includes the m -order seasonal differencing to reduce the seasonal component. Please refer to [41] for the details of adding seasonal differencing terms for the SARIMA model.

Below, the STL method [42] is introduced, where STL stands for “seasonal-trend decomposition based on LOESS” and LOESS stands for “locally estimated scatterplot smoothing”. The STL method decomposes a time series into three components, the trend, seasonality, and remainder. A trend shows a persistent increasing or decreasing tendency in data, seasonality indicates seasonal effects over a fixed period of time, and the remainder corresponds to the data noise. Specifically, the STL method runs iteration by iteration to decompose time series by finding the data trend and seasonal data trend with the help of the smoother LOESS [51]. The basic idea of LOESS is that a function can be well fitted in a small neighborhood by a low-order polynomial. LOESS runs fast and fits data properly, which in turn makes STL run fast and well.

One can rely on the STL method to decompose a time series and focus on the remainder for accurately spotting anomalies in the time series, as shown in [44]. As also shown in [44], one can rely on the SARIMA model to formulate a time series, either stationary or non-stationary, for spotting anomalies in the time series. By combining the STL method and the SARIMA model, the ADSaS method can achieve good anomaly detection performance. The ADSaS method was proposed in [44], where ADSaS stands for “anomaly detection with SARIMA and STL”. The method can collectively well consider four performance criteria of the anomaly detection system, which are accuracy, speed, the model size, and domain universality. SARIMA can perform time-series forecast accurately, and STL can perform time-series decomposition well. As the ADSaS method integrates SARIMA and STL, it has good anomaly detection performance.

2.2. Pearson Correlation Coefficient

The Pearson correlation coefficient (PCC), or Pearson product-moment correlation coefficient (PPMCC), was proposed by Pearson in 1895 [36] for measuring the linear correlation between two variables X and Y . Let X and Y be two time series. The PCC of X and Y , denoted by $Cor(X, Y)$ or $\rho_{X,Y}$, is formulated according to (6).

$$Cor(X, Y) = \frac{Cov(X, Y)}{\sigma_X \sigma_Y}, \quad (6)$$

where $Cov(X, Y)$ is the covariance of X and Y , σ_X is the standard deviation of X , and σ_Y is the standard deviation of Y .

$Cor(X, Y)$ can be used to measure the linear correlation between X and Y . Its value ranges between $+1$ and -1 , where $+1$ indicates the totally positive linear correlation, -1 indicates the totally negative correlation, and 0 indicates no linear correlation between X and Y .

2.3. Dickey–Fuller Test

Dickey and Fuller proposed the Dickey–Fuller (DF) test or the unit root test in 1979 [37] for checking the stationarity of time series on the basis of the auto-regressive model. As mentioned earlier, a time series is stationary if its mean, variance, and covariance do not vary with time. The DF test has the null hypothesis that a unit root is present in the auto-regressive model of a time series. Note that if a unit root exists for an auto-regression model of a time series, then the series is non-stationary. For example, let the simple AR(1) auto-regressive model of a time series be $x_t = a_1 x_{t-1} + e_t$. The time series is non-stationary if $a_1 \geq 1$; on the contrary, it is stationary if $a_1 < 1$. Note that a unit root is present if $a_1 = 1$. Therefore, the DL test is under the null hypothesis that a_1 is equal to 1 (or the model has the root equal to the unity) and the alternative hypothesis that $a_1 < 1$ (or the model has the root not equal to the unity). In summary, if the null hypothesis is rejected, then the time series is stationary; otherwise, the time series is non-stationary.

Below, in this paper, the DF test is executed by setting the p -value threshold to be 0.0005, as taken by the ADSaS method [44]. When the p -value returned by the test is less than the threshold of 0.0005, the null hypothesis is rejected and the time series is assumed to be stationary; otherwise, the null hypothesis is accepted and the time series is regarded as non-stationary.

2.4. Discret Wavelet Transform

Wavelet transform (WT) utilizes wavelets as the basis for decomposing signals or time series, where wavelets are functions that are defined over a finite interval of time and whose average value is zero [38]. Wavelets are also called mother wavelets. They are exemplified by Haar, Meyer, Morlet, and Daubechies, among others. WT is a very well-known and useful tool for extracting the time–frequency features of a time series.

Let $\psi(t)$ be a mother wavelet. The child wavelet $\psi_{a,b}(t)$ of $\psi(t)$ is defined according to (7).

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}} + \left(\frac{t-b}{a} \right) \quad (7)$$

where a is the dilation (scaling) parameter and b is the transition (shifting) parameter. High-scaled child wavelets correspond to lower frequencies and are used as low-pass filters. On the contrary, low-scaled child wavelets correspond to higher frequencies and are used as high-pass filters.

WT can be divided into two categories, the discrete wavelet transform (DWT) and the continuous wavelet transform (CWT). Below, the DWT, which is adopted by this research, is introduced. As shown in Figure 1, the DWT consists of high-pass filters (HPFs) and low-pass filters (LPFs), both along with down sampling filters (DSFs). Time series data first go through high-pass filters to generate detail coefficients (denoted as cD) to represent high-frequency components of data. Moreover, time series data go through low-pass filters to generate approximation coefficients (denoted as cA) to represent low-frequency components of data. Afterwards, detail coefficients are retained, whereas approximation coefficients are further decomposed again into high- and low-frequency components until a specific number of decompositions is reached. Finally, all retained detail coefficients and the last approximation coefficients altogether constitute the result of the DWT.

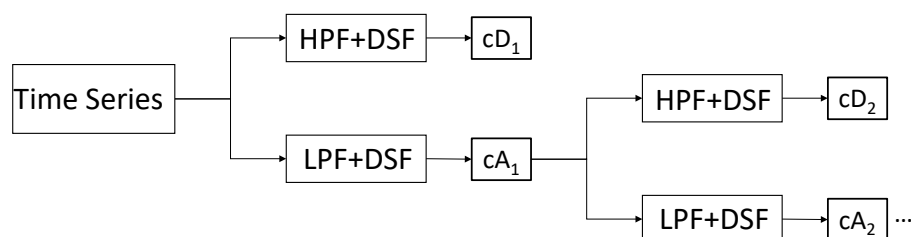


Figure 1. The process of the discrete wavelet transform.

2.5. Autoencoder

The autoencoder (AE) [39] is a special type of artificial neural network (ANN) whose input and output (or label) are identical. An ANN consists of layers of artificial neurons, each of which can take input vector x , compute and output $f(wx^T + b)$, where $f(\cdot)$ is an activation function (e.g., a hyperbolic tangent function), w is a weight vector, x^T is the transposition of x , and b is a bias vector. Figure 2 illustrates the architecture of a simple AE. As shown in Figure 2, the AE has the encoder, code, and decoder parts. It is divided into front layers, a middle layer, and rear layers. The front layers can be regarded as the encoder, whereas the rear layers as the decoder. The middle layer constitutes the code, which can be used as a latent representation of the input. The middle layer associated with the code usually has much fewer neurons than the input layer. An AE can thus be used to reduce the dimensionality of input data. Specifically, the code of the AE can be regarded as the dimensionality reduction result or feature extraction result of the input data.

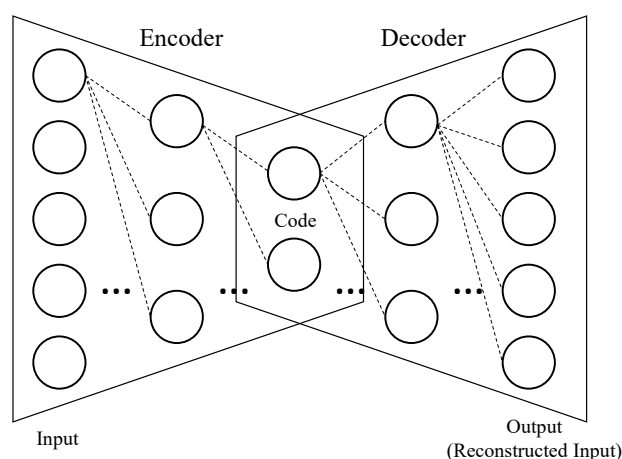


Figure 2. The architecture of an autoencoder.

In general, an AE constructs a code by encoding the input into the code. It reconstructs the input by decoding the code back to the input. The difference between the input and the reconstructed input is called the reconstruction error. Well-known methods, such as error backpropagation and gradient descent, and different optimizers like adaptive moment estimation (Adam), can be used to optimize (or minimize) reconstruction errors.

3. Proposed Anomaly Detection Framework

The proposed anomaly detection framework, Tri-CAD, for univariate time series is introduced in this section. As shown in Figure 3, Tri-CAD first classifies time series into three classes: (Class 1) periodic, (Class 2) stationary, and (Class 3) non-periodic and non-stationary time series. The classification is based on the Pearson product-moment correlation coefficient [36] and Dickey–Fuller test [37]. The threshold settings in Figure 3 for the Pearson product-moment correlation coefficient ρ and the Dickey–Fuller test p -value will be described later. The first D (say, $D = 1000$) data are assumed to be normal and are used for the classification. Below, in this paper, the set of the first D data is called the normal dataset or the training dataset, whereas the set of the rest of the data is called the test dataset.

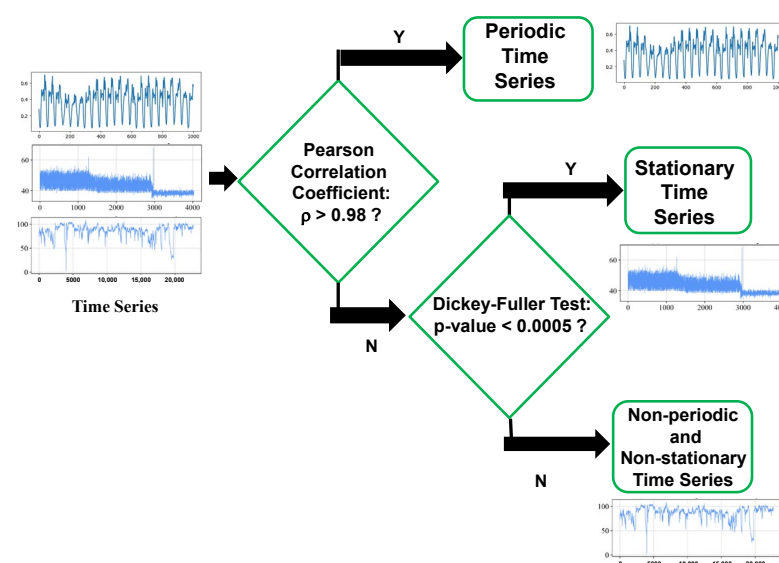


Figure 3. The proposed anomaly detection framework.

Tri-CAD first relies on the normal dataset to divide time series into three classes. Then, different schemes that use statistics, wavelet transform, and the autoencoder are applied to

different classes of time series. The schemes are applied to the normal dataset to generate statistic and deep learning features of the time series. Afterwards, the same schemes are applied to the test dataset to generate statistic and deep learning features. Finally, the concept of standard deviation is applied for anomaly detection. Specifically, if the statistic and deep learning features of the normal dataset and the test dataset have a large difference that exceeds a certain threshold, defined by the standard deviation concept, anomalies are assumed to occur. How Tri-CAD performs the classification and anomaly detection is described below for each class of time series.

3.1. Periodic Time Series Anomaly Detection

Tri-CAD adopts the Pearson correlation coefficient for checking whether a time series is periodic. As shown in Figure 4, two consecutive equal-sized sub-time series Y and Z of size w , $w \leq D/2$, are first derived from the D data of the normal dataset. Next, the Pearson product-moment correlation coefficient $Cor(Y, Z)$ is derived for any possible value of w . The maximum $Cor(Y, Z)$ value ρ and the corresponding value w are then obtained. If the value ρ is greater than a pre-specified threshold (e.g., 0.98), the time series is considered to be periodic with the period of w .

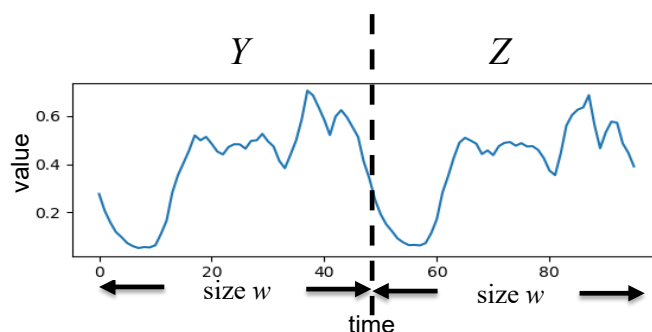


Figure 4. Illustration of using the Pearson correlation coefficient of two consecutive equal-sized time windows to check the periodicity of time series.

When the time series is determined to be a periodic time series with the period w , a sliding time window of size w is applied to the D data of the normal dataset, and the skewness of every time window of data is derived. Skewness is a measure of the symmetry of a distribution. The skewness of a time series X of n data is defined according to (8).

$$\gamma(X) = \frac{n}{(n-1)(n-2)} \sum_{x \in X} \left(\frac{x - \mu_X}{\sigma_X} \right)^3 \quad (8)$$

where μ_X and σ_X are the mean and standard deviation of X , respectively. Tri-CAD adopts skewness for detecting anomaly because skewness is sensitive to the removal of outliers (anomalies), as shown in [52]. After obtaining the skewness values of all sliding windows for the D data of the normal dataset, the moving averages of k pervious skewness values are derived. Finally, the mean μ and the standard deviation σ of the moving averages of skewness values are derived. Then, a similar process is applied to the test dataset. Specifically, a sliding time window of size w is also applied to all data in the test dataset, and the skewness value of every time window of data is derived. For a test data point x , the moving average of k pervious skewness values is derived and is regarded as its anomaly score s_x . Its standard anomaly score (SAS) λ_x is then calculated according to (9).

$$\lambda_x = \frac{|s_x - \mu|}{\sigma} \quad (9)$$

Note that the SAS λ_x of a data point x is the measurement of how far x 's anomaly score s_x deviates from μ in terms of σ . As shown earlier, a different λ_x value corresponds

to different occurrence probability of such a data point. For example, $\lambda_x = 4.417173$ corresponds to 0.001% of occurrence. If the SAS λ_x exceeds a pre-specified deviation threshold τ and λ_x is increasing (i.e., x 's SAS is larger than the SAS of x 's previous data point), then the data point x is assumed to be anomalous.

3.2. Stationary Time Series Anomaly Detection

After Tri-CAD judges a time series not to be periodic, it employs the Dickey–Fuller test [37] with the p -value threshold of 0.0005, which is also taken in [44], to check whether a time series is stationary. This is achieved under the null hypothesis that a unit root exists in an autoregressive model of the time series. If the p -value returned by a test is less than the threshold of 0.0005, the time series is assumed to be stationary. As mentioned earlier, the D data of the normal dataset are taken for the Dickey–Fuller test.

When the time series is determined to be stationary, Tri-CAD calculates the means of the data within a large global sliding time window and a small local sliding time window, as shown in Figure 5. The global window and the local window have the same ending time point. Furthermore, the size (say, 100) of the global window is much larger than the size (say, 5) of the local window. For a data point x , the means of its global window and local window are first calculated, which are denoted as $Gmean_x$ and $Lmean_x$, respectively. Then, x 's anomaly score s_x is calculated according to (10).

$$s_x = \left| \frac{Gmean_x - Lmean_x}{Gmean_x} \right| \quad (10)$$

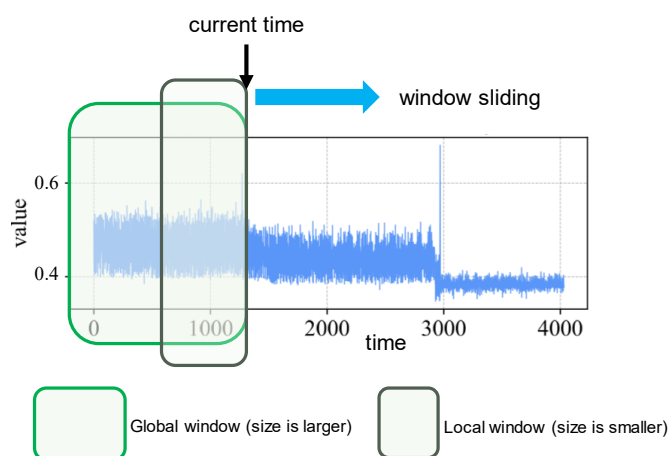


Figure 5. A large global sliding time window and a small local sliding time window of a stationary time series for calculating the ratio of data means.

The anomaly scores of the D data in the normal dataset and their mean and standard deviation are calculated. Afterwards, for a test data point x with the anomaly score s_x , its SAS λ_x is calculated according to Equation (9). Similarly, if λ_x is increasing and exceeds a pre-specified threshold, the data point x is regarded as anomalous.

3.3. Non-Periodic and Non-Stationary Time Series Anomaly Detection

A time series is regarded to be non-periodic and non-stationary if it is judged to be neither periodic nor stationary. For non-periodic and non-stationary time series, Tri-CAD adopts WT to extract the time–frequency features of the time series and then uses the AE to encode and then decode the time–frequency features. For WT and AE to execute properly, the standardization mechanism is used to standardize the original time series data so that the data follow the standard normal distribution with the mean of 0 and the

standard deviation of 1. The standardization is performed by transforming each data value according to (11).

$$x'_i = \frac{x_i - \mu}{\sigma} \quad (11)$$

where x_i is the original data value; x'_i is the standardized data value; and μ and σ are the mean and the standard deviation of the original data, respectively.

Tri-CAD applies DWT and AE to the D data of the normal dataset. To be more specific, a moving sliding time window of size ws (say $ws = 60$) is used to get ws data to go through DWT with Haar as the mother wavelet to get ws DWT coefficients. The ws coefficients are then taken as the input to train the AE model. The stride of the moving sliding time window is 1. So, there are $D - ws + 1$ sliding windows of data taken as the training data to train the AE model. The reconstruction errors of the AE are obtained, and the mean μ and the standard deviation σ of the errors are calculated.

The AE neural network structure is shown in Figure 6. The sizes of the input layer and the output layer are both ws (say, 60). The numbers of neurons in the hidden layers are 32, 16, 8, 4, 2, 4, 8, 16, and 32, respectively. The code has the dimension of 2, as the number of neurons in the middle layer is 2. The encoder takes the scaled exponential linear unit (SELU) function as the activation function, whereas the decoder takes the hyperbolic tangent (Tanh) function as the activation function. The error function (or loss function) is the mean square error (MSE), and the optimizer is the adaptive moment estimation (Adam). To avoid overfitting, the early stop mechanism is used. Specifically, a part of the training dataset is split out to be the validation data set. The training process of the AE model proceeds epoch by epoch. Once the MSE value of the validation data starts to rise, the training stops, so that the AE model does not fit the training data too closely to avoid the overfitting problem.

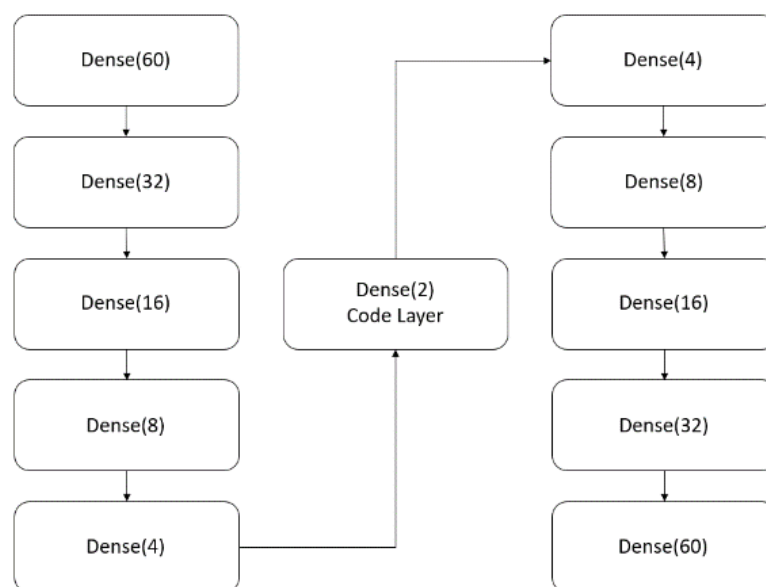


Figure 6. The AE neural network structure.

Similarly, Tri-CAD applies WT and AE to every data point x in the test dataset. The AE reconstruction error is assumed to be the anomaly score of x , and the SAS λ_x is derived according to Equation (9). Likewise, if λ_x is increasing and exceeds a pre-specified threshold, the data point x is regarded as anomalous.

3.4. Threshold Setting

As mentioned earlier, Tri-CAD is a semi-supervised anomaly detection framework and thus does not rely on labels to train models for detecting anomalies. Likewise, Tri-CAD

does not depend on labels to set the threshold of detecting anomalies. Tri-CAD actually assumes that the first D data are normal, and derives the mean μ and the standard deviation σ of certain features of the D data. It then checks if the standard anomaly score λ_x of a test data point x exceeds a certain deviation threshold τ (i.e., if $\lambda_x > \tau \times \sigma$). If so, the data point x is assumed to be anomalous; otherwise, x is assumed to be normal.

The deviation threshold τ plays the role of making the anomaly detection more sensitive or less sensitive. We can thus set different levels of sensitivity based on the deviation threshold τ . For example, setting τ to be 2.575829, 3.290527, 3.890592, 4.417173, 4.891638, 5.326724, 5.730729, and 6.109410 can make the framework have eight sensitivity levels of anomaly detection, so that 1%, 0.1%, 0.01%, 0.001%, 0.0001%, 0.00001%, 0.000001%, and 0.0000001% of data are detected as anomalies. Certainly, if the framework can obtain some data labels in advance, then the deviation threshold τ can be set as an optimized value to make the anomaly detection have the best performance.

3.5. Predecessors of Tri-CAD

As mentioned earlier, the proposed Tri-CAD framework is similar to the framework proposed in [45]. However, the two frameworks are different. Specifically, Tri-CAD first checks whether a time series is periodic. It then checks whether the time series is stationary or non-stationary. On the contrary, the framework proposed in [45] first checks if a time series is stationary. It then checks whether or not the time series is periodic. Checking that a time series is periodic can be done more definitely and clearly than checking that a time series is stationary. Hence, Tri-CAD can achieve more definite classification of time series than the framework proposed in [45].

As also mentioned earlier, Tri-CAD integrates an anomaly detection mechanism, WAAD, proposed in [46]. WAAD is used for detecting anomalies in non-periodic and non-stationary time series. It first applies the DWT to the time series of a sliding time window to obtain wavelet transform coefficients. It then uses an AE to encode and decode (reconstruct) these coefficients. WAAD detects anomalies by monitoring the reconstruction error between the original and reconstructed coefficients for every time window. An anomaly is assumed to occur if the error meets specific conditions. In Tri-CAD, WAAD is improved by replacing the reconstruction error with the standard anomaly score. If the standard anomaly score is increasing and exceeds a pre-specified threshold, then an anomaly is assumed to occur. In practice, the improved WAAD has better performance than the original WAAD. It also has the merits that the standard anomaly score threshold value can be set intuitively.

4. Performance Evaluation and Comparison

The NAB (Numenta anomaly benchmark) [40] is used for evaluating the prediction performance of the proposed Tri-CAD framework. NAB contains many labeled time series datasets, which are either artificial or real-world time series. The datasets have entries containing timestamps and values. The timestamp is of the format “yyyy-mm-dd hh.mm.ss” representing the year, month, day, hour, minute, and second. The difference of adjacent timestamps is fixed; however, such a difference varies for various datasets. An entry is labeled as anomalous if its timestamp is specified explicitly in a separate file. As Tri-CAD is for semi-supervised anomaly detection, it does not rely on labels to train anomaly detection models. The labels are used only for the purpose of performance evaluation.

This paper applies three real-world time series for performance evaluation. The applied time series are (a) “NAB NYC taxi”, the data of taxi passenger counts in the New York city; (b) “NAB CPU Utilization”, the CPU utilization data from Amazon Web Services (AWS); and (c) “NAB Machine Temperature”, the data from a temperature sensor attached to a machine component.

The performance measurements are precision, recall, and F_1 -score of the prediction, whose definitions are shown in (12)–(14), respectively. In the definitions, TP (true positive) represents the number of anomalies that are truly (or correctly) detected as anomalies,

FP (false positive) represents the number of normal data that are falsely (or incorrectly) detected as anomalies, *TN* (true negative) represents the number of normal data that are truly detected as normal data, and *FN* (false negative) represents the number of anomalies that are falsely detected as normal data.

$$precision = TP / (TP + FP) \quad (12)$$

$$recall = TP / (TP + FN) \quad (13)$$

$$F_1\text{-score} = 2 \times precision / (precision + recall) \quad (14)$$

The NAB time series have labels given on the basis of anomaly windows [40]. Specifically, the entire time series is divided into time windows of a fixed window size (*fws*). Note that different NAB datasets have different window sizes. If one or more anomalies appear in a time window, the time window is labeled as an anomaly window; otherwise, it is labeled as a normal window. Note that a labeled anomaly corresponds to an anomaly window whose center is the labeled anomaly.

Figures 7–14 show the performance evaluation results of Tri-CAD using the three NAB datasets with the deviation threshold set as 3.89, 8, and 8.35, respectively. The performance evaluation is also conducted for different training dataset sizes and different WT sliding time window sizes. Specifically, the NYC taxi dataset and the CPU utilization dataset are assumed to have training datasets with a small size of 500 or a large size of 1000 (i.e., $D = 500$ or 1000). The machine temperature dataset is assumed to have training datasets with a small size of 1000 or a large size of 2000 (i.e., $D = 1000$ or 2000). Moreover, the machine temperature dataset is also assumed to have WT sliding window sizes of a small size of 30 or a large size of 60 (i.e., $ws = 30$ or 60). In Figures 7–14, the anomaly windows are marked with pink strips, whereas detected anomalies are marked with red dots. If anomalies are claimed to be detected in an anomaly window by the proposed framework, then *TP* is increased by one. On the contrary, if anomalies are claimed to be detected in a normal window, then *FP* is increased by one. Note that all anomalies in a time window altogether increase a count by one. Similarly, if no anomaly is claimed to be detected in an anomaly window, then *FN* is increased by one. After *TP*, *FP*, and *FN* are derived, the precision, recall, and F_1 -score can be evaluated accordingly.

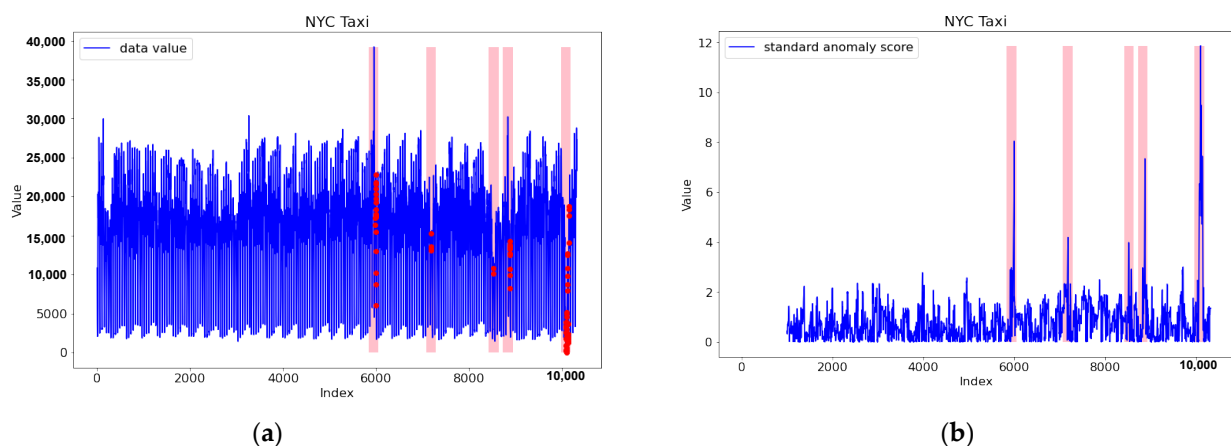


Figure 7. Tri-CAD performance evaluation using NAB NCY Taxi dataset with training dataset size $D = 500$: (a) the time series and anomaly points and (b) standard anomaly scores and anomaly windows.

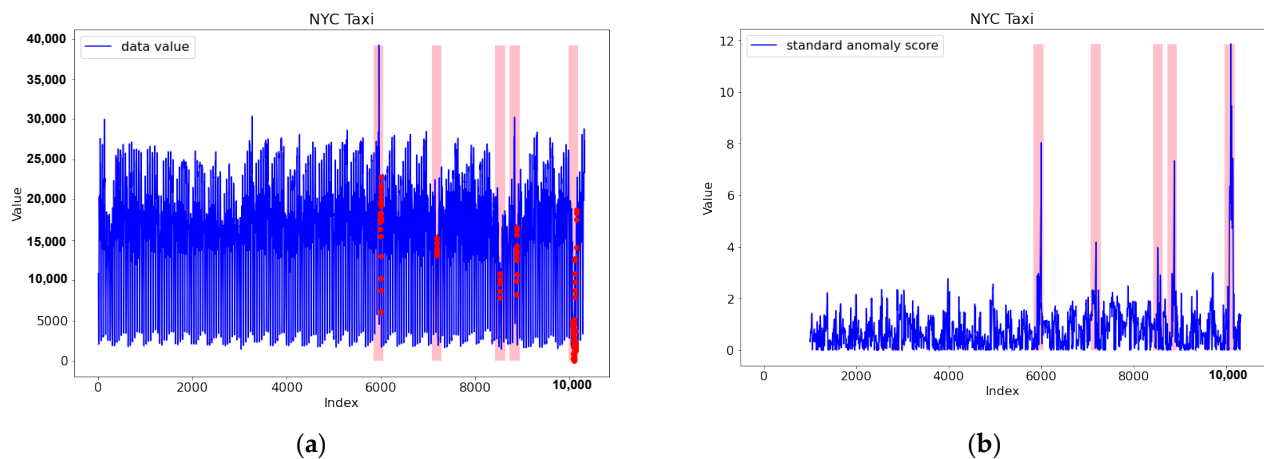


Figure 8. Tri-CAD performance evaluation using NAB NYC Taxi dataset with training dataset size $D = 1000$: (a) the time series and anomaly points and (b) standard anomaly scores and anomaly windows.

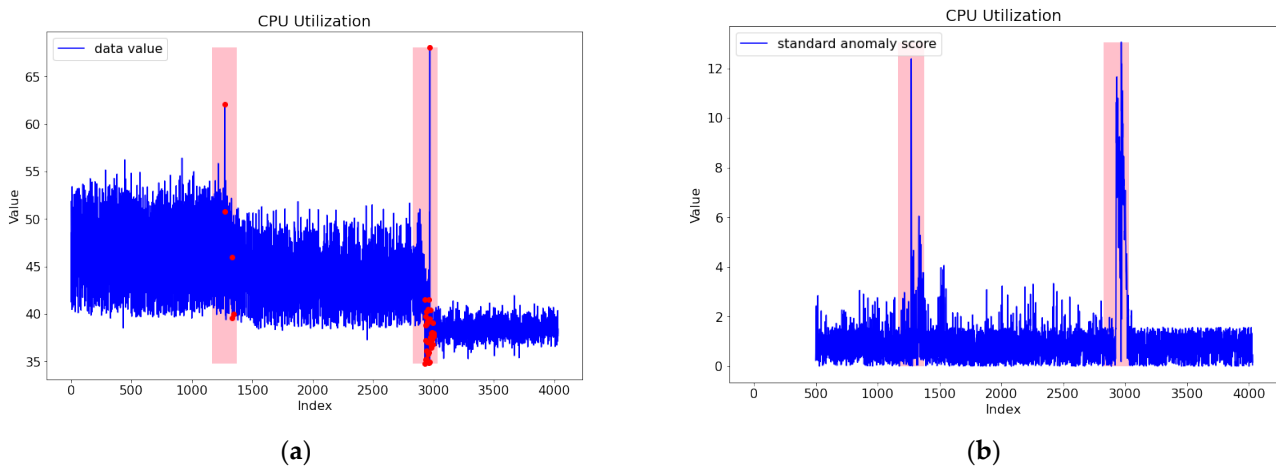


Figure 9. Tri-CAD performance evaluation using NAB AWS CPU Utilization dataset with training dataset size $D = 500$: (a) the time series and anomaly points and (b) standard anomaly scores and anomaly windows.

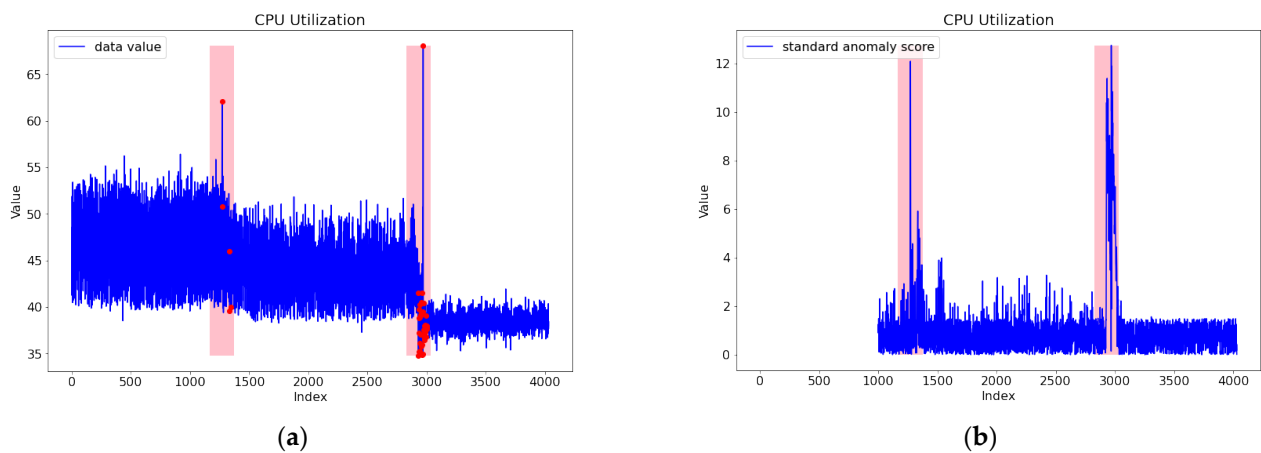


Figure 10. Tri-CAD performance evaluation using NAB AWS CPU Utilization dataset with training dataset size $D = 1000$: (a) the time series and anomaly points and (b) standard anomaly scores and anomaly windows.

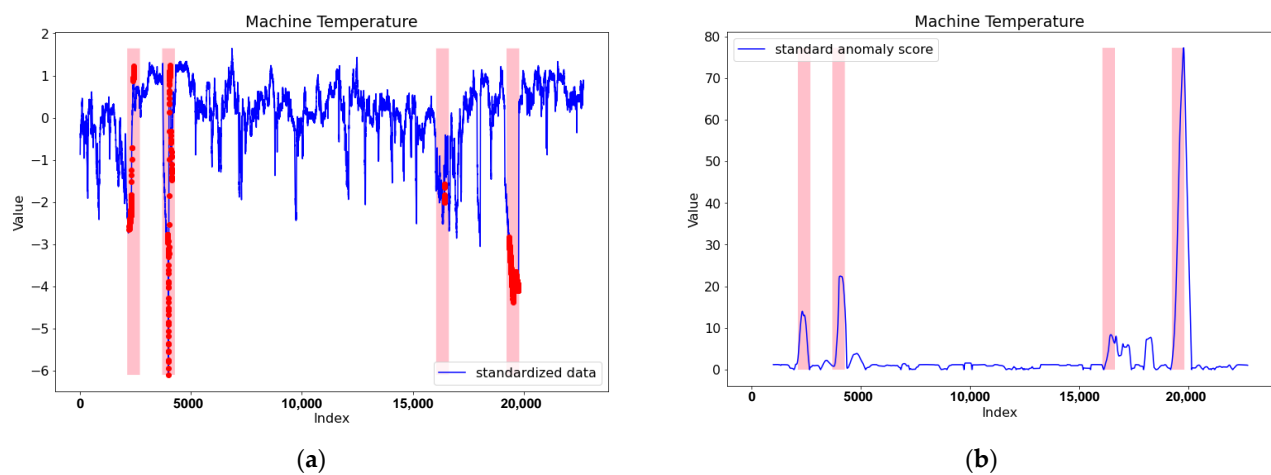


Figure 11. Tri-CAD performance evaluation using NAB Machine Temperature dataset with training dataset size $D = 1000$ and WT window size $ws = 30$: (a) the standardized time series and anomaly points and (b) standard anomaly scores and anomaly windows.

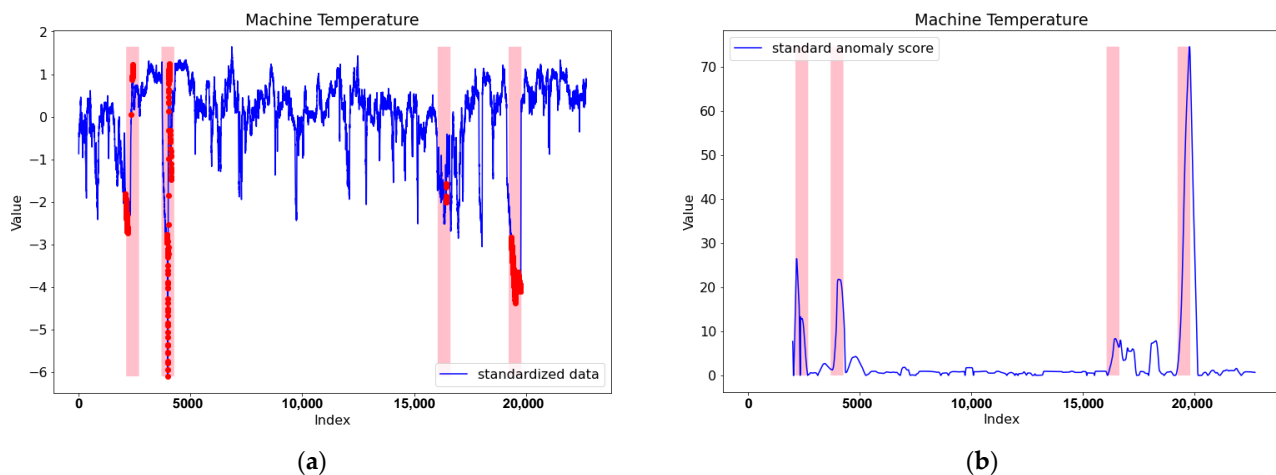


Figure 12. Tri-CAD performance evaluation using NAB Machine Temperature dataset with training dataset size $D = 2000$ and WT window size $ws = 30$: (a) the standardized time series and anomaly points and (b) standard anomaly scores and anomaly windows.

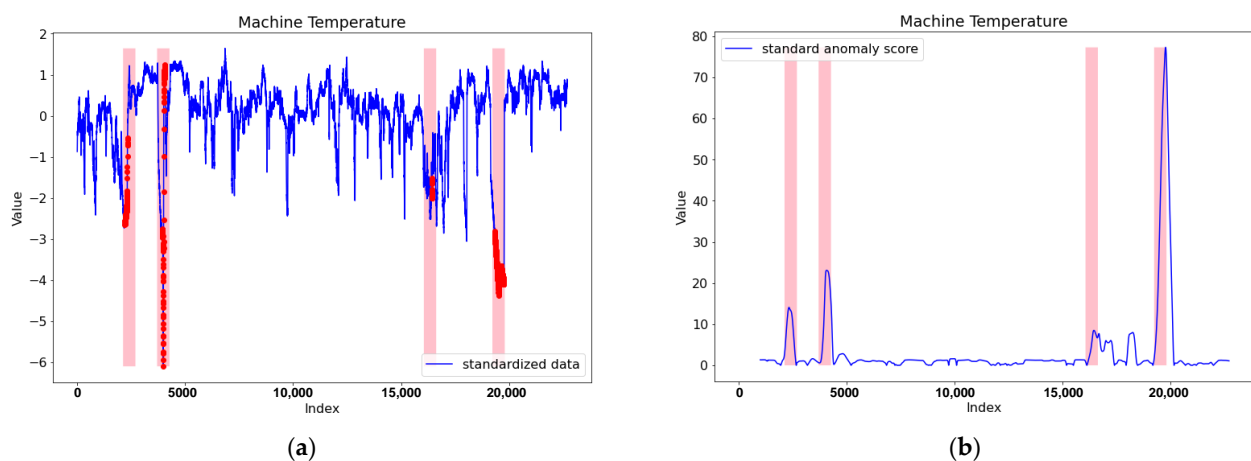


Figure 13. Tri-CAD performance evaluation using NAB Machine Temperature dataset with training dataset size $D = 1000$ and WT window size $ws = 60$: (a) the standardized time series and anomaly points and (b) standard anomaly scores and anomaly windows.

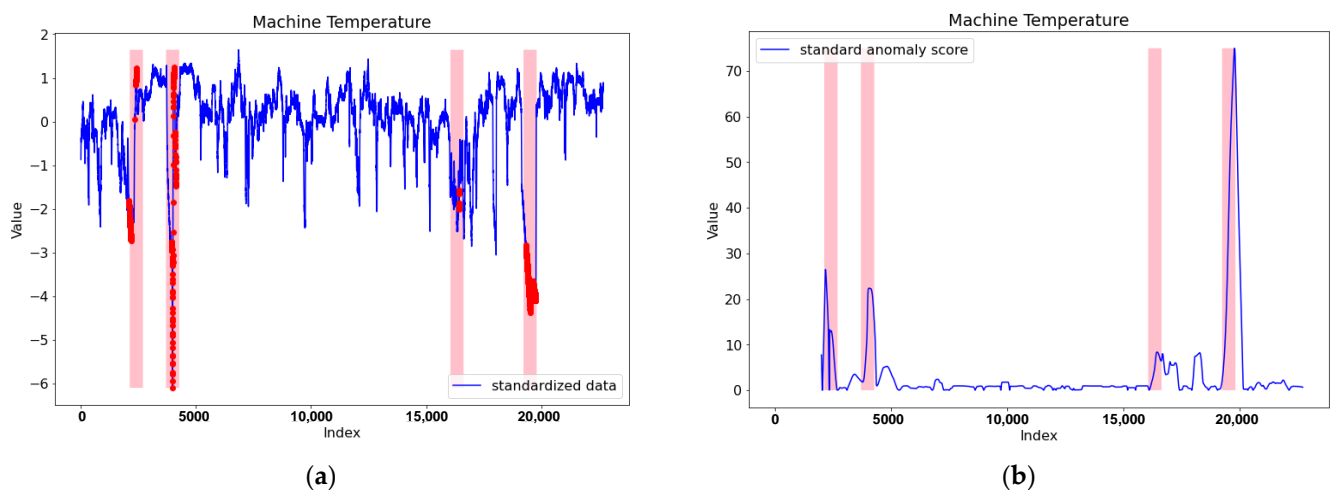


Figure 14. Tri-CAD performance evaluation using NAB Machine Temperature dataset with training dataset size $D = 2000$ and WT window size $ws = 60$: (a) the standardized time series and anomaly points and (b) standard anomaly scores and anomaly windows.

It can be observed from Figures 7–14 that Tri-CAD has similar performance results for different training dataset sizes and different WT sliding window sizes. Tri-CAD has the best performance (i.e., 1.0) in terms of the precision, recall, and F_1 -score either for small or large training dataset sizes. It also has best performance (i.e., 1.0) either for small or large sliding window sizes. However, when the window sizes are not within the range between the small value and the large value, Tri-CAD does not work with high performance. Certainly, such window sizes should be avoided and their associated figures are not shown in this paper to save space.

The evaluation results of the proposed framework Tri-CAD are compared with those of related methods, namely the STL, SARIMA, LSTM, LSTM with STL, and ADSaS. Note that the related methods may have anomaly window sizes that are different from those set by NBA datasets. The performance comparisons for the three NAB time series are shown in Table 1, where the best performance measurements are shown in boldface. Note that the “NAB NYC Taxi” dataset is periodic, the “NAB CPU” dataset is stationary, and the “NAB Machine Temperature” dataset is non-periodic and non-stationary. By Table 1, we can see that the proposed framework Tri-CAD has the best performance in all measurements like the precision, recall, and F_1 -score for all datasets. This is because related methods try to improve anomaly detection performance by integrating various models, each of which is suitable for a different class of time series. On the contrary, the proposed framework first classifies time series into different classes and then utilizes a specific scheme to perform anomaly detection for a specific class of time series. As it is more likely to perform better by applying a special scheme to a special class of time series, the proposed framework thus outperforms others.

Table 1. Comparisons of the proposed framework Tri-CAD and related methods.

Time Series	Class	Fixed Window Size (<i>fw</i> s)	Metrics	STL only	SARIMA only	LSTM only	LSTM with STL	ADSaS	Proposed Framework Tri-CAD
NAB	Class 1	206	Precision	0.533	0.000	0.176	0.161	1.000	1.000
NYC			Recall	0.889	0.000	0.333	1.000	1.000	1.000
Taxi			F_1 -score	0.667	0.000	0.231	0.277	1.000	1.000
NAB	Class 2	200	Precision	0.800	0.143	0.833	0.308	1.000	1.000
CPU			Recall	1.000	0.250	1.000	1.000	0.250	1.000
Utilization			F_1 -score	0.889	0.182	0.909	0.471	0.400	1.000
NAB	Class 3	566	Precision	0.250	0.000	0.049	0.059	1.000	1.000
Machine			Recall	0.222	0.000	0.222	0.625	0.500	1.000
Temperature			F_1 -score	0.235	0.000	0.080	0.108	0.667	1.000

Time series Class 1: periodic; Class 2: stationary; Class 3: non-periodic and non-stationary.

5. Conclusions

This paper proposes a semi-supervised anomaly detection framework, called Tri-CAD, for univariate time series with statistics and autoencoder deep learning technologies. The Pearson correlation coefficient and the Dickey–Fuller test are used to classify time series into three classes: the periodic, the stationary, and the non-periodic and non-stationary time series. Different schemes are then applied to different classes of time series for performing anomaly detection properly.

Three NAB datasets are applied to performance evaluation of the proposed Tri-CAD framework for different training dataset sizes and different WT sliding time window sizes. It is observed that Tri-CAD has the best performance of 1 in terms of the precision, recall, and F_1 -score either for the small or the large training dataset sizes. It also has the best performance of 1 either for the small or the large WT sliding window sizes. The evaluated performance results of Tri-CAD are compared with those of related methods, namely, the STL, SARIMA, LSTM, LSTM with STL, and ADSaS. The comparisons show that Tri-CAD outperforms the others in terms of the precision, recall, and F_1 -score for all three datasets.

In the future, we plan to employ a nonparametric Bayesian approach [53] and/or a method using online clustering [54] to set the anomaly detection threshold values. We also plan to apply the proposed framework to more datasets to achieve more comprehensive performance evaluation. Furthermore, we will try to apply the design concept of the proposed univariate time series anomaly detection framework to multivariate time series that are much complex than univariate time series.

Author Contributions: J.-R.J., J.-B.K., and Y.-L.L. together designed the proposed framework. J.-B.K. and Y.-L.L. implemented the framework and carried out performance evaluation for it. J.-R.J. wrote the paper reporting the background, related work, the proposed framework, and the performance evaluation and comparisons of the framework and other related methods. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Ministry of Science and Technology (MOST), Taiwan, under the grant number 109-2622-E-008-028-.

Conflicts of Interest: The authors declare no conflict of interest.

References

- Chandola, V.; Banerjee, A.; Kumar, V. Anomaly detection: A survey. *ACM Comput. Surv.* **2009**, *41*, 1–58. [\[CrossRef\]](#)
- Chalapathy, R.; Chawla, S. Deep learning for anomaly detection: A survey. *arXiv* **2019**, arXiv:1901.03407.
- Zhang, J. Advancements of outlier detection: A survey. *ICST Trans. Scalable Inf. Syst.* **2013**, *13*, 1–26. [\[CrossRef\]](#)
- Gupta, M.; Gao, J.; Aggarwal, C.C.; Han, J. Outlier detection for temporal data: A survey. *IEEE Trans. Knowl. Data Eng.* **2013**, *26*, 2250–2267. [\[CrossRef\]](#)
- Pimentel, M.A.; Clifton, D.A.; Clifton, L.; Tarassenko, L. A review of novelty detection. *Signal Process.* **2014**, *99*, 215–249. [\[CrossRef\]](#)

6. Zhao, Z.; Mehrotra, K.G.; Mohan, C.K. Online anomaly detection using random forest. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*; Springer: Cham, Switzerland, June 2018; pp. 135–147.
7. Beckman, R.J.; Cook, R.D. Outlier s. *Technometrics* **1983**, *25*, 119–149.
8. Tripathi, A.; Yadav, D.R. Outlier detection in big data using optimized Orion technique. *Int. J. Curr. Res. Life Sci.* **2018**, *7*, 1204–1207.
9. Al-khatib, A.A.; Balfaqih, M.; Khelil, A. A survey on outlier detection in Internet of Things big data. In *Big Data-Enabled Internet of Things*; IET Digital Library: London, UK, 2019; Chapter 11; pp. 225–272.
10. Shahid, N.; Naqvi, I.H.; Qaisar, S.B. Characteristics and classification of outlier detection techniques for wireless sensor networks in harsh environments: A survey. *Artif. Intell. Rev.* **2015**, *43*, 193–228. [CrossRef]
11. Van, N.T.; Tinh, T.N. An anomaly-based network intrusion detection system using deep learning. In Proceedings of the 2017 International Conference on System Science and Engineering (ICSSE), Ho Chi Minh City, Vietnam, 21–23 July 2017; pp. 210–214.
12. Maimó, L.F.; Gómez, Á.L.P.; Clemente, F.J.G.; Pérez, M.G.; Pérez, G.M. A self-adaptive deep learning-based system for anomaly detection in 5G networks. *IEEE Access* **2018**, *6*, 7700–7712. [CrossRef]
13. Lu, N.; Du, Q.; Sun, L.; Ren, P. Traffic-driven intrusion detection for massive MTC towards 5G networks. In Proceedings of the IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Honolulu, HI, USA, 15–19 April 2018; pp. 426–431.
14. Choi, H.; Kim, M.; Lee, G.; Kim, W. Unsupervised learning approach for network intrusion detection system using autoencoders. *J. Supercomput.* **2019**, *75*, 5597–5621. [CrossRef]
15. Choi, S.; Youm, S.; Kang, Y.S. Development of scalable on-line anomaly detection system for autonomous and adaptive manufacturing processes. *Appl. Sci.* **2019**, *9*, 4502. [CrossRef]
16. Wang, P.; Yang, L.T.; Nie, X.; Ren, Z.; Li, J.; Kuang, L. Data-driven software defined network attack detection: State-of-the-art and perspectives. *Inf. Sci.* **2020**, *513*, 65–83. [CrossRef]
17. Krishnaveni, S.; Vigneshwar, P.; Kishore, S.; Jothi, B.; Sivamohan, S. Anomaly-based intrusion detection system using support vector machine. In *Artificial Intelligence and Evolutionary Computations in Engineering Systems*; Springer: Singapore, 2020; pp. 723–731.
18. Kubota, T.; Yamamoto, W. Anomaly detection from online monitoring of system operations using recurrent neural network. *Procedia Manuf.* **2019**, *30*, 83–89. [CrossRef]
19. Lindemann, B.; Fesenmayr, F.; Jazdi, N.; Weyrich, M. Anomaly detection in discrete manufacturing using self-learning approaches. *Procedia CIRP* **2019**, *79*, 313–318. [CrossRef]
20. Hsieh, R.J.; Chou, J.; Ho, C.H. Unsupervised online anomaly detection on multivariate sensing time series data for smart manufacturing. In Proceedings of the 2019 IEEE 12th Conference on Service-Oriented Computing and Applications (SOCA), Kaohsiung, Taiwan, 18–21 November 2019; pp. 90–97.
21. Wu, J.; Zhao, Z.; Sun, C.; Yan, R.; Chen, X. Fault-attention generative probabilistic adversarial autoencoder for machine anomaly detection. *IEEE Trans. Ind. Informatics* **2020**, *16*, 7479–7488. [CrossRef]
22. Nawaratne, R.; Alahakoon, D.; De Silva, D.; Yu, X. Spatiotemporal anomaly detection using deep learning for real-time video surveillance. *IEEE Trans. Ind. Inform.* **2019**, *16*, 393–402. [CrossRef]
23. Singh, G.; Kapoor, R.; Khosla, A.K. Intelligent anomaly detection video surveillance systems for smart cities. In *Driving the Development, Management, and Sustainability of Cognitive Cities*; IGI Global: Hershey, PA, USA, 2019; pp. 163–182.
24. Di Mauro, N.; Ferilli, S. Unsupervised LSTMs-based learning for anomaly detection in highway traffic data. In *International Symposium on Methodologies for Intelligent Systems*; Springer: Cham, Switzerland, October 2018; pp. 281–290.
25. Xu, M.; Wu, J.; Wang, H.; Cao, M. Anomaly detection in road networks using sliding-window tensor factorization. *IEEE Trans. Intell. Transp. Syst.* **2019**, *20*, 4704–4713. [CrossRef]
26. Malini, N.; Pushpa, M. Analysis on credit card fraud identification techniques based on KNN and outlier detection. In Proceedings of the 2017 Third International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB), Chennai, India, 27–28 February 2017; pp. 255–258.
27. Popat, R.R.; Chaudhary, J. A survey on credit card fraud detection using machine learning. In Proceedings of the 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 11–12 May 2018; pp. 1120–1125.
28. Sheshasaayee, A.; Thomas, S.S. A purview of the impact of supervised learning methodologies on health insurance fraud detection. In *Information Systems Design and Intelligent Applications*; Springer: Singapore, 2018; pp. 978–984.
29. Thill, M.; Däubener, S.; Konen, W.; Bäck, T. Anomaly detection in electrocardiogram readings with stacked LSTM networks. In Proceedings of the 19th Conference Information Technologies-Applications and Theory (ITAT 2019), Donovaly, Slovakia, 20–24 September 2019; pp. 17–25.
30. Rastatter, S.; Moe, T.; Gangopadhyay, A.; Weaver, A. Abnormal Traffic Pattern Detection in Real-Time Financial Transactions (No 827). Available online: https://easychair.org/publications/preprint_open/CKvQ (accessed on 21 July 2021).
31. Tam, N.T.; Weidlich, M.; Zheng, B.; Yin, H.; Hung NQ, V.; Stantic, B. From anomaly detection to rumour detection using data streams of social platforms. *Proc. VLDB Endow.* **2019**, *12*, 1016–1029. [CrossRef]
32. Wu, H.S. A survey of research on anomaly detection for time series. In Proceedings of the 2016 13th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), Chengdu, China, 16–18 December 2016; pp. 426–431.

33. Islam, M.R.; Sultana, N.; Moni, M.A.; Sarkar, P.C.; Rahman, B. A comprehensive survey of time series anomaly detection in online social network data. *Int. J. Comput. Appl.* **2017**, *180*, 13–22.
34. Cook, A.; Misirli, G.; Fan, Z. Anomaly detection for IoT time-series data: A survey. *IEEE Internet Things J.* **2019**, *7*, 6481–6494. [[CrossRef](#)]
35. Blázquez-García, A.; Conde, A.; Mori, U.; Lozano, J.A. A review on outlier/anomaly detection in time series data. *arXiv* **2019**, arXiv:2002.04236.
36. Pearson, K. Notes on regression and inheritance in the case of two parents. *Proc. R. Soc. Lond.* **1895**, *58*, 240–242.
37. Dickey, D.A.; Fuller, W.A. Distribution of the estimators for autoregressive time series with a unit root. *J. Am. Stat. Assoc.* **1979**, *74*, 427–431.
38. Chui, C.K. *An Introduction to Wavelets*; Elsevier: Amsterdam, The Netherlands, 2016.
39. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning Internal Representations by Error Propagation. Available online: <https://apps.dtic.mil/sti/pdfs/ADA164453.pdf> (accessed on 21 July 2021).
40. Numenta Anomaly Benchmark. Available online: <https://github.com/numenta/NAB> (accessed on 15 March 2020).
41. Xu, S.; Chan, H.K.; Zhang, T. Forecasting the demand of the aviation industry using hybrid time series SARIMA-SVR approach. *Transp. Res. Part E Logist. Transp. Rev.* **2019**, *122*, 169–180. [[CrossRef](#)]
42. Cleveland, R.; Cleveland, W.S.; McRae, J.E.; Terpenning, I.J. STL: A seasonal-trend decomposition procedure based on loess (with discussion). *J. Off. Stat.* **1990**, *6*, 3–73.
43. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [[CrossRef](#)]
44. Lee, S.; Kim, H.K. ADSaS: Comprehensive real-time anomaly detection system. In *International Workshop on Information Security Applications*; Springer: Cham, Switzerland, 2018; pp. 29–41.
45. Kao, J.B.; Jiang, J.R. Anomaly detection for univariate time series with statistics and deep learning. In Proceedings of the 2019 IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE), Yunlin, Taiwan, 3–6 October 2019; pp. 404–407.
46. Li, Y.L.; Jiang, J.R. Anomaly detection for non-stationary and non-periodic univariate time series. In Proceedings of the 2020 IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE), Yunlin, Taiwan, 23–25 October 2020; pp. 177–179.
47. Ruppert, D.; Matteson, D.S. Time series models: Basics. In *Statistics and Data Analysis for Financial Engineering*; Springer: New York, NY, USA, 2015; pp. 307–360.
48. Konasani, V.R.; Kadre, S. Time-series analysis and forecasting. In *Practical Business Analytics Using SAS*; Apress: Berkeley, CA, USA, 2015; pp. 441–507.
49. Box, G.E.P.; Jenkins, G.M. Some comments on a paper by Chatfield and Prothero and on a review by Kendall. *J. R. Stat. Society. Ser. A* **1973**, *136*, 337–352. [[CrossRef](#)]
50. Konasani, V.R.; Kadre, S. Multiple regression analysis. In *Practical Business Analytics Using SAS*; Apress: Berkeley, CA, USA, 2015; pp. 351–400.
51. Cleveland, W.S.; Loader, C. Smoothing by local regression: Principles and methods. In *Statistical Theory and Computational Aspects of Smoothing*; Physica-Verlag HD: Heidelberg, Germany, 1996; pp. 10–49.
52. Heymann, S.; Latapy, M.; Magnien, C. Outskewer: Using skewness to spot outliers in samples and time series. In Proceedings of the 2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, Istanbul, Turkey, 26–29 August 2012; pp. 527–534.
53. Alhakami, W.; Alharbi, A.; Bourouis, S.; Alroobaea, R.; Bouguila, N. Network anomaly intrusion detection using a nonparametric Bayesian approach and feature selection. *IEEE Access* **2019**, *7*, 52181–52190. [[CrossRef](#)]
54. Hosseini, A.; Sarrafzadeh, M. Unsupervised prediction of negative health events ahead of time. In Proceedings of the 2019 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI), Chicago, IL, USA, 19–22 May 2019; pp. 1–4.