IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 16, 2021, accepted July 27, 2021, date of publication August 2, 2021, date of current version August 10, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3101844

A Novel Multivariate Time-Series Anomaly Detection Approach Using an Unsupervised Deep Neural Network

PEIHAI ZHAO[1], (Member, IEEE), XIAOYAN CHANG[1], AND MIMI WANG[2,3], (Member, IEEE)
[1]School of Computer Science and Technology, Donghua University, Shanghai 201620, China
[2]College of Information Science and Technology, Donghua University, Shanghai 201620, China
[3]Engineering Research Center of Digitized Textile and Fashion Technology, Ministry of Education, Donghua University, Shanghai 201620, China

Corresponding author: Mimi Wang (wangmimi2013@hotmail.com)

This work was supported in part by the Fundamental Research Funds for the Central Universities under Grant 2232021D-21 and Grant 2232020D-51, and in part by the Shanghai Municipal Commission of Economy and Information Civil-Military Inosculation Project under Grant JMRH-2018-1042.

ABSTRACT With the development of hardware technology, we can collect increasingly reliable time series data, in which time series anomaly detection is an important task to find problems in time and avoid risks. It is not easy to establish a multivariate time series anomaly detection system, because the collected data not only have different attributes, scales, and characteristic information but also have horizontal and vertical connections among these data collected by various sensors. In addition, there is no clear boundary regarding whether the data are abnormal, and there is currently no unified definition of multidimensional time-series anomalies. Recently, deep learning methods have shown outstanding advantages in the processing of multidimensional time series. In this paper, we propose a definition of point anomalies in multivariate time series and an unsupervised deep learning method, the multilayer convolutional recurrent autoencoded anomaly detector (MCRAAD), which is used to detect anomalies in multivariate time series. We calculate the feature matrix sequence through the data in the sliding window, extract the characteristics of the feature matrix sequence by a multilayer convolutional encoder, obtain the time relations in the feature matrix by using several ConvLSTM units, and finally reconstruct the feature matrix sequence with the convolutional decoder to predict the self-feature matrix. In addition, we propose a threshold setting method to assist with the determination of anomalies. Finally, we test our model on synthetic datasets and a real dataset of house monitoring. The results of experiments show that our method is superior to these basic models in detecting ability and robustness. This model also provides an effective method for multivariate time-series anomaly detection in real life.

INDEX TERMS Anomaly detection, multivariate time-series, deep neural network, unsupervised learning.

I. INTRODUCTION

There are many time-series data in the real world such as stock prices, weather data, personal health data, and sensor data. One of these important fields of time-series is anomaly detection. Time-series anomaly detection has very important significance and has become a necessary part of the modern manufacturing industry and information services, because undetected anomalies may cause serious damage [1]. Currently, there are many studies on abnormal detection of time-series data for specific scenarios. For example, Gupta *et al.* [2] studied abnormal changes in GDP components over time, and Derya Birant and Alp Kut [3] identified abnormal weather from the wave heights of the four seas. Keogh *et al.* [4] checked whether an electrocardiogram had abnormal fluctuations. Although many researchers have performed many studies on time-series anomalies, they are still ambiguous.

At present, there is no unified definition of time-series anomalies [1], and they are defined as unusual patterns that do

The associate editor coordinating the review of this manuscript and approving it for publication was Zijian Zhang.

not conform to expected behavior. Anomaly can be described as a data point that is significantly dissimilar to other data points in [5]. Anomalies can be classified into three categories [6]: point, contextual and collective. Point anomalies refer to when the data of a single instance are abnormal compared to other data, which can occur in any type of data. A contextual anomaly means that the data are abnormal in a specific scenario but normal in another scenario, which can only occur in relative data. Time is a context attribute in a time-series that determines the position of instances in the entire sequence. A collective anomaly means that there may be no correlation among multiple individual instances, but they may be abnormal as a group. In this paper, we take the contextual attributes and behavioral attributes of each data instance into consideration and define a point anomaly of multivariate time-series based on the relationship of a single instance and its historical data. Anomaly detection of multidimensional time-series is inseparable from multivariate time-series data.

In recent years, due to the development of industry and the Internet of Things [7], multivariate time-series anomaly detection technology has made great progress. We can obtain more reliable time-series data from the devices by configuring a multisensor system. However, processing these data from sensors is a major problem. First, the data that are collected by different sensors may have different attributes, frequencies, and dependencies. Therefore, the preprocessing of these data is a very time-consuming task and may require some domain knowledge. Jin *et al.* [8] proposed an innovative learning framework for multi-variate air pollutant concentration prediction. This method, which separated the features and trends by decomposing the original data into high-frequency part and low-frequency part to learn them respectively in a multi-channel module, provided a great idea for us to acquire the features of multivariate time series. In addition to the problems mentioned above, there are still some unavoidable problems; for example, it is difficult to set an accurate boundary for normal and abnormal data, or the data collected by different sensors may contain noise due to other factors. These data with serious noise may look similar to anomalies [1], lead to false alarms [9], and affect the performance of algorithms [10]. The fact that the amount of normal data is much larger than the amount of abnormal data is another problem, and the problem of extremely unbalanced data has become another major trouble spot in time-series anomaly detection [11]. Researchers have tried many ways to process multivariate time-series data.

To establish an automatic detection system of anomalies in time-series, many researchers have proposed many effective models and methods to deal with time-series data. Many scholars have been studying time-series modeling including ARIMA [12], SVM [13], and CNN [1]. Since the data used for anomaly detection usually have no clear labels and the amount of abnormal data is very small, many unsupervised discriminative approaches are used for anomaly detection, including OCSVM [14], iForest [15], and LSTM-ED [16]. Although these unsupervised methods have made some progress in the field of time-series anomaly detection, many models still cannot detect anomalies effectively. We summarize several reasons. First, there is a close time dependence between multidimensional time-series data, and general density-based methods and clustering models cannot capture the dependence between series. Second, multivariate time-series from the real world containing relatively severe noise may reduce the generalization ability of the detection model. Anomaly scores are generally used to measure the severity of anomalies and find the root cause of the anomaly. The existing methods for finding the root cause of the anomaly, e.g., RCA [17], will also fail to accurately find the root cause due to noise. Finally, the problem of data imbalance will cause the model to be unable to fully obtain the relationship between normal data and abnormal data, which will lead to a poor detection effect.

Taking all the problems mentioned above into consideration, we propose an unsupervised deep neural network named the Multilayer Convolutional Recurrent Autoencoded Anomaly Detector (MCRAAD) to detect anomalies at certain time steps in multidimensional time-series. Specifically, MCRAAD first obtains the intercorrelation between series by a window-based feature matrix sequence and then uses convolutional neural networks to extract the characteristics of the time-series. We take advantage of a convolutional long short-term memory network to capture the temporal patterns, reconstruct the feature matrix through a convolutional decoder, predict the normal data according to the reconstructed feature matrix sequence and compare it with the real data to determine whether an anomaly event has occurred. Ideally, when there is an anomaly, MCRAAD cannot predict the current situation well. If the distance between the predicted data and the real data is greater than a certain threshold, it will be considered that there may be an abnormal event. Figure 1 (B) shows the difference between the actual data and the forecast data by MCRAAD under normal and abnormal conditions. The closer the color is to blue, the smaller the difference; while the closer the color is to yellow, the greater the difference. The part enclosed with blue braces in (A) is the normal time-series data collected under normal conditions. MCRAAD can predict its self-feature matrix well, and the distinction between the self-feature matrix and the real one is small, as shown on both sides of (B). The data enclosed in red braces is the abnormal data. The difference matrix at some time step of this period can be demonstrated in the middle of (B), which has more parts that are close to yellow.

The main contributions of our work are as follows:

- A novel framework, MCRAAD, is proposed for the anomaly detection of multivariate time-series. Sliding-window-based cross-correlation computation is adopted to transform multivariate time-series into feature matrices. Multilayer CNNs can capture the hidden nonlinear cross-correlation features of multivariate time-series

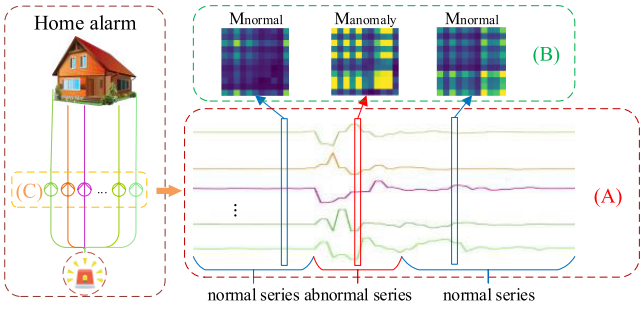


FIGURE 1. Housing emergency alarming system. (A) denotes multivariate time-series data, (B) shows the difference between data predicted by MCRAAD and real data at certain time steps in both normal and abnormal situations, and (C) represents a system for collecting multivariate time-series data.

with no need of a priori knowledge, and predict its own feature matrix to determine abnormal events.

- We propose a novel threshold setting strategy of anomaly scores using a normal training set, which will help determine abnormal events in multivariate time-series.
- We perform extensive experiments on synthetic datasets and a real home monitoring dataset. The results of our method demonstrate that MCRAAD is superior to other models in terms of anomaly detection performance and robustness.

The paper is structured as follows: section II analyzes related work, and section III describes the proposed definition and method in detail. The experiments and discussions are detailed in section IV. Finally, the conclusions and future works are listed in section V.

II. RELATED WORK

A time-series is sequenced set of data on a metric for each interval of time in chronological order. Data in a sequence are related to the previous data. This type of data can describe the changes and reflect the development of things. A time-series X can be formally defined as $X = \{x^1, x^2, x^3, \cdots, x^n\}$, where n is the number of observed time steps [18], x^1 is the result of the first time step observation, and x^2 is the recording result after a certain period of time behind the first observation. Similarly, x^n is the value of the last time step. Time-series data can be divided into different categories according to different aspects.

Time-series data can be classified into two types according to the number of variables: univariate time-series and multivariate time-series. A univariate time-series only needs to consider the tendency of the observed value of one variable over time. However, the relationships between the various series also need to be considered in multivariate time series. The observed value for k monitoring variables at time t $(1 \leq t \leq n)$ is $x^t = \{x_1^t, x_2^t, x_3^t, \cdots, x_k^t\}$, where $x^t \in X$. x_1^t is the observation of the first variable at time t, similarly, x_k^t is the value of the k-th variable at the same time step. Generally, we store the time-series in the form of a matrix, and $X = \{x^1, x^2, x^3, \cdots, x^n\} = (x_1, x_2, x_3, \cdots, x_k)^T \in \mathbb{R}^{k \times n}$ can be regarded as a k-dimensional vector. The univariate time-series is a one-dimensional $(k = 1)$ vector, while the multivariate time-series is a k-dimensional $(k > 1)$ vector whose length equals to n. The size of $X = (x_1, x_2, x_3, \cdots, x_k)^T$ is $k \times n$, and the value $x^t = (x_1^t, x_2^t, x_3^t, \cdots, x_k^t)$ is a k-dimensional vector representing the observed value at time t. Different kinds of time-series data have different priorities to consider.

It is necessary to consider the correlation between the series when dealing with multivariate time-series data. Relevant features may produce noise and mask the true anomalies [19]. Correlation analysis based on a sliding window is an important method of preprocessing multidimensional data [20]. This method divides sequences into many overlapping subsequences and then calculates the correlations among the subsequence data in a window by a given algorithm. Chen *et al.* [21] proposed a sliding window convolutional differential autoencoder that can detect the anomalies of multivariate time-series in time and space. When processing multidimensional time-series, we also need to consider the characteristics of the data itself in addition to the correlation between series.

There are many time-series data in the real world, but only a small part of the data has been clearly labeled as abnormal or not. It is difficult to design a clear boundary between normal and abnormal data because they are relative and related to the context in which the data are located. In addition, the amount of abnormal data in a whole sequence is very small compared with the normal data. Therefore, many researchers have proposed unsupervised methods of anomaly detection. One of these popular techniques is clustering methods, e.g., One-Class Support Vector Machine (OCSVM). Ma and Perkins [14] used OCSVM to model the training data and distinguished the test data as normal or abnormal. Although this method can solve the problem effectively in many applications, it does not perform well on time-series data because the method can only pay attention to the information of the data itself but has nothing to do with capturing the correlation of temporal data. Another approach is machine learning methods, e.g., Isolation Forest (iForest). Liu *et al.* [15] proposed a method based on iForest to detect anomalies in sequence data. This approach establishes a binary tree by random selection of features and cannot obtain the relationship between various features, which leads to an unstable detection effect. Some researchers considered using predictive methods to capture temporal information. These predictive models can predict what data will appear at the next time step before new data arrive. Some representative models are the Autoregressive model (AR) [22], Moving Average (MA) [23], Autoregressive Moving Average (ARMA) [24], and its variants. These models have shown good performance in the field of time-series anomaly detection, but they are easily affected by noise. Thus, there will be many false negative and false positive results when data have serious noise. In recent years, anomaly detection

methods based on deep learning have made great progress. One of the representative deep learning models for processing time-series is Long Short-Term Memory (LSTM). An LSTM-based model can predict the data at the next time step and calculate the distance between predicted data and real time-series data to determine whether it is abnormal. Ergen and Kozat [25] proposed a model based on the LSTM structure to detect product quality. Lindemann *et al.* [26] used the LSTM structure to obtain the features of these fixed-length sequences and established an anomaly detection model based on OCSVM. Li *et al.* [27] combined the characteristics of SAE and LSTM and proposed an anomaly detection method based on unsupervised deep learning. The LSTM structure, which can learn the nonlinear relationship of short-term or long-term time-series data, is also sensitive to noise, which may increase the risk of misclassification in anomaly detection. Liang *et al.* [9] calculated the feature matrices and added a forgetting mechanism to the model to alleviate the influence of noise. Zhang *et al.* [28] proposed a model combining wavelet denoising and principal component analysis to process data noise. The autoencoder model is another popular noise-friendly model used for detecting sequence data anomalies. It judges whether it is abnormal by considering the difference between encoded data and original data. Vincent *et al.* [29] built a self-encoding noise reduction model that can restore the input data with noise to the data without noise. Borghesi *et al.* [30] used a semi-supervised anomaly detection method based on an autoencoder. The autoencoder uses the idea of encoding and decoding to suppress the noise in time-series, which makes the model more robust, but the autoencoder is theoretically better at dealing with one-dimensional time-series [9] because this structure is unable to obtain the correlation between multidimensional time series data. AnoGAN [31] is the first framework presented in unsupervised anomaly detection, and successfully detects diseased images from sets of unknown images. Plakias [32] proposed GANs based one-class fault detection model for the multi-dimensional problem and experiments indicated the proposed method outperforms One-class SVM and Isolation Forest. As stated, GANs based architecture has been gradually developed in anomaly detection fields. The main drawback of GAN architecture is the instability during training [33]. Combining the advantages of the above models, the MCRAAD model we proposed can not only capture the time pattern effectively but also has the ability to deal with noise. Although all of these methods based on a variety of technologies have different effects on time-series anomaly detection, they all need to have indicators to judge anomalies in common.

An important aspect of any anomaly detection technique is the manner of reporting anomalies. Typically, the outputs produced by anomaly detection techniques are two types [6]: scores and labels. The scoring technique assigns an anomaly score for each instance in the test data to evaluate the abnormal degree of each instance. Generally, a threshold is set to determine whether it is abnormal. Zhang *et al.* [34] used the anomaly score to determine whether it is abnormal. Lin *et al.* [35] used labels that can directly mark the abnormal state of the data to distinguish normal and abnormal events. Due to the continuity and relevance of time-series data, the abnormal state of the data is closely related to the context, and it is not easy to distinguish whether it is abnormal. Therefore, it is more appropriate to calculate the score at each time step to measure the abnormal state of this time step. The anomaly score can measure the degree to which the observed value at time step t conforms to the current trend. The smaller the anomaly score, the closer it is to the current temporal pattern. Conversely, the larger the anomaly score, the further away it is from the temporal characteristics. To help us judge abnormal events with abnormal scores, we also propose a method to set the threshold of abnormal scores with a normal training set.

III. ANOMALY DEFINITION AND MODEL DETAILS

In this section, we introduce the problem we aim to study in section III-A, and then define the concept of multivariate time-series point anomalies in section III-B. Next, we elaborate on the proposed MCRAAD in detail in section III-C. Specifically, we show how to generate feature matrices. Then, we encode the spatial information in feature matrices via a convolutional encoder and model the temporal pattern with ConvLSTM. Finally, we reconstruct feature matrices based upon a convolutional decoder and predict self-feature matrices with these reconstructed feature matrices. We introduce in detail the strategy of threshold setting and the method of calculating anomaly scores in section III-D.

A. PROBLEM STATEMENT

Given the historical data of k time-series with length n, $X = (x_1, x_2, x_3, \cdots, x_k)^T \in \mathbb{R}^{k\times n}$, and assuming there exists no anomaly in the data, we aim to detect anomalous events at certain time steps after n. We only use the normal dataset for training to characterize the various patterns of sensors under normal conditions. At testing time, we determine whether the current state is normal or abnormal by means of the anomaly score.

B. ANOMALY DEFINITION OF TIME-SERIES

1) DEFINITION 1: SUBSEQUENCE OF TIME-SERIES

A subsequence $S_{sub} \in \mathbb{R}^{k\times z}$ of a time-series X is a continuous subset of the values from X of length z starting from position t. Formally, $S_{sub}, z = \left\{x^t, x^{t+1}, \cdots, x^{t+z-1}\right\}$.

2) DEFINITION 2: POINT ANOMALY OF TIME-SERIES

Let $X = \left\{x^1, x^2, x^3, \cdots, x^n\right\}$ be a time-series, Y be a set, $z \in N$ be a natural number, and $\delta \in R$ be a real number. $\exists y \in Y$ meets the mapping $g : (X, S_{sub}) \to Y$ for all $x^t \in X$ and $S^t_{sub} = \left\{x^{t-z}, x^{t-z+1}, \cdots, x^{t-1}\right\}$ $(t = 1, 2, \cdots, n)$, as $y = g(x^t, S^t_{sub})$. The state of step t is

$$\begin{cases} 0\,(normal), & g(S^t_{sub}, x^t) \leqslant \delta \\ 1\,(anomaly), & g(S^t_{sub}, x^t) > \delta. \end{cases}$$

TABLE 1. An example of a multivariatel time-series anomalies.

Time	1	2	3	4	5	6	7	8	9	10
Data	-2	-3	-2	-1	**1**	0	0	1	-1	1
	1	3	5	3	**-2**	2	1	0	-1	2
	-4	-2	-3	-5	**-2**	-6	-7	-6	-5	-8
z	3									
δ	10									
$g(S^t_{sub}, x^t)$	$g(S^t_{sub}, x^t) = \begin{cases} 0, t <= z \\ \frac{\sum_{b=1}^{z}\sum_{a=1}^{k}(x_a^{t-b} - x_b^t)^2}{z \times k}, t > z \end{cases}$									
y	0	0	0	3	**15**	6	5	3	3	4
Nomal(0/1)	0	0	0	0	**1**	0	0	0	0	0

*The red column is the abnormal position of the time-series

To make the concept of anomalies easier to understand, we take an example of multidimensional time-series anomaly. Table 1 shows an example of a 3-dimensional time-series with one point anomaly. The multivariate time-series data with three series of length 10 has an unusual step at time 5 marked with red color. The table also displays us $z = 3$, $\delta = 10$, and the mapping formulation g. When $t \leq z$, $y = 0$; and $t > z$, the value of y can be calculated by g. For example, when $t = 5$, $S^5_{sub} = \begin{bmatrix} -3 & -2 & -1 \\ 3 & 5 & 3 \\ -2 & -3 & -5 \end{bmatrix}$, $x^5 = (1, -2, -2)^T$, the value of this step $y = \frac{(-3-1)^2 + (3-(-2))^2 + \ldots + (-5-(-2))^2}{3 \times 3} = 15$ is greater than δ which equals to 10. Thus, we will study whether the event at time step 5 is an abnormal event. We perform the same operation for the other time steps to obtain the corresponding value between x^t and its preceding subsequence S^t_{sub}. For example, the value at the ninth time step equal to 3 is less than δ, which shows that it is a normal event.

C. DETAILS OF MCRAAD FRAMEWORK

Many of the proposed time-series point anomaly detection methods attempt to find the mapping relation that is consistent with the original series. We also propose a framework to find mapping relationships and detect unusual events in multivariate time-series. MCRAAD consists of four parts: generating feature matrices is introduced in section III-C1, the convolutional encoder is displayed in section III-C2, middle-layer ConvLSTM will be presented in section III-C3 and convolutional decoder is shown in section III-C4. The framework of MCRAAD can be abstracted as shown in Figure 2. Part (A) shows multilayer full convolutional encoders extracting the features of the input feature matrix sequence, (B) is the ConvLSTM layer to capture the current temporal pattern, and (C) denotes multilayer convolutional decoders that have the same size as the encoders. The output of (C) is the reconstructed characteristic matrix, which is the input of a linear structure predicting the self-feature matrix. (S) demonstrates the multivariate time-series data, and Loss Function calculates the loss of real self-feature matrices and predicted self-feature matrices.

1) GENERATE FEATURE MATRICES

It is significant to study the correlation of time-series when researching multivariate time-series [36]. Hallac *et al.* [37] suggested that the correlations between different pairs of time-series are critical to characterize the system status. One of the methods is to calculate the feature matrix of the overlapping subsequence in the sliding window to reflect the correlation. We follow the method of calculating the feature matrix proposed in [9] to amplify features and reduce noise. To detect whether the observed value of time t in sequence is consistent with the current temporal trend, we need to calculate two characteristic matrices at time t in the time-series X. One is the feature matrix, which is calculated with a subsequence of length w before the time step t (including time t), where w is the size of the sliding window. We denote this type of matrix as M^t, which reflects the correlation characteristics of the subsequence of length w before time t. Another feature map $\mathcal{M}^t$ is only interested in the information of the observed value at time t. The algorithm generating the feature matrix sequence is shown as Algorithm 1. In this algorithm, we first visit the data of the entire time-series sequentially. Then, at each visit, the current data and a subsequence before the current data are extracted to calculate the self-feature matrix and feature matrix, respectively, according to Eq. 1. Finally, we obtain the feature matrix sequence and self-feature matrix sequence by saving the two kinds of feature matrices at each visit in order.

The more detailed steps for calculating the two kinds of feature matrices are as follows. When setting $w = 1$, the result of this self-feature map only contains its own information. When $w > 1$, we obtain the feature map that contains the relationship of this time step t and the previous subsequence of time t with length w. We need subsequence $Y^t_{sub} = \{x^{t-w+1}, x^{t-w+2}, \cdots, x^t\}$ with length w selected from time t forward to generate the feature matrix at time t, where w is the window size. Y^t_{subs} can be regarded as a matrix of $k \times w$, and the transposed matrix of Y^t_{subs} is a matrix of $w \times k$. The feature matrix M^t at time t can be calculated by multiplying the matrix Y^t_{sub} by the transpose matrix of Y^t_{sub} and dividing by the window size w. M^t is formulated as:

$$M^t = \frac{Y^t_{sub}\left(Y^t_{sub}\right)^T}{w} \tag{1}$$

The value M^t_{pq} at this position of the p-th row and the q-th column of the feature matrix M^t at time t is calculated as follows:

$$M^t_{pq} = \frac{\sum_{\alpha=0}^{w-1} x_i^{t-\alpha} \times x_j^{t-\alpha}}{w} \tag{2}$$

where $x_i = \{x_i^{t-w+1}, x_i^{t-w+2}, \cdots, x_i^t\}$, $x_j = \{x_j^{t-w+1}, x_j^{t-w+2}, \cdots, x_j^t\}$, w is the size of sliding window, $i = p, j = q$, and M^t is a matrix of $k \times k$. When calculating

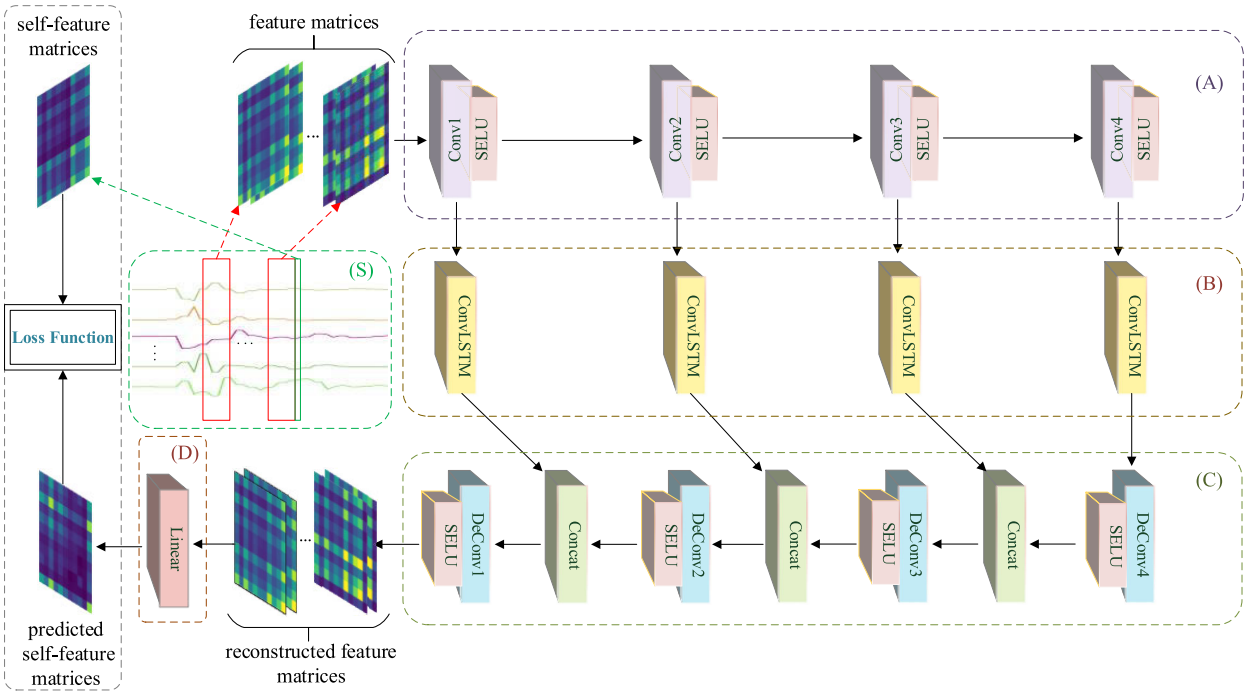


FIGURE 2. Overall framework of model MCRAAD. Multiple full convolutional encoders in (A) extract features of input feature matrix sequence. Input (A) is feature matrix subsequence, (B) is ConvLSTM layer, and number of convolutional decoders in (C) is equal to encoder. Output of (C) is reconstructed characteristic matrix, (D) is one-layer linear structure predicting self-feature matrix, (S) is multivariate time-series data, and loss function calculates loss of real self-feature matrices and predicted self-feature matrices.

Algorithm 1 Generate Feature Matrices Algorithm

Input: Multivariate time-series X, lenth of the time-series n, dimension of the time-series k, window size w.

Output: feature matrix sequence M, seft-feature matrix sequence $\mathcal{M}$.

for int $t = 1$ **to** n **do**
  if $t \leq w$ **then** $M[t] \leftarrow 0 \in \mathbb{R}^{k\times k}$
  else
    $S_{sub} \leftarrow [x[t-w+1], x[t-w+2], \cdots, x[t]]$
    $M[t] \leftarrow \dfrac{S_{sub} \times S_{sub}^T}{w}$
  $\mathcal{M}[t] \leftarrow x[t] \times x[t]^T$
return $M, \mathcal{M}$

its self-feature matrix at this step, we should set $w = 1$ and then obtain a matrix $Y_{sub}^t = \{x^t\} = (x_1^t, x_2^t, x_3^t, \cdots, x_k^t)$ of $k \times 1$. Similarly, we can calculate the self-feature matrix $\mathcal{M}^t$ that only contains the current time step information by inputting Y_{sub}^t into Eq. 1 and Eq. 2. Moving the same window one step to the next time step, we will obtain the subsequence Y_{sub}^{t+1} at the next step and obtain two kinds of feature matrices at time $t+1$. By repeating the above steps, we can obtain the feature matrices of the entire sequence. It should be noted that the subsequence of length w cannot be fetched from time t forward when $t < w$. Thus, we directly use the zero matrices of $k \times k$ as the feature map at time t.

The characteristic matrix sequence generated by the entire sequence is marked as M, and the self-feature matrix sequence generated by the entire sequence containing only its own information is marked as $\mathcal{M}$. Figure 2 (S) is the data at one time step. The feature matrix calculated using only the information at this time step is the part indicated by the green dashed arrow. The red box adjacent to the green box takes a subsequence of the time-series data, and this subsequence can be calculated into a feature matrix that contains the information of the previous segment of the subsequence adjacent to the green box. We also list an example of computing the two kinds of feature matrices at time step 5 with the time-series data listed in Table 1.

For example, taking a subsequence of length $w = 3$ from Table 1 at time step $t = 5$ as $Y_{sub}^5 = \begin{bmatrix} -2 & -1 & 1 \\ 5 & 3 & -2 \\ -3 & -5 & -2 \end{bmatrix}$, $Y_{sub}^{5T} = \begin{bmatrix} -2 & 5 & -3 \\ -1 & 3 & -5 \\ 1 & -2 & -2 \end{bmatrix}$, the feature matrix of this step is $M^5 = \dfrac{\begin{bmatrix} -2 & -1 & 1 \\ 5 & 3 & -2 \\ -3 & -5 & -2 \end{bmatrix} \times \begin{bmatrix} -2 & 5 & -3 \\ -1 & 3 & -5 \\ 1 & -2 & -2 \end{bmatrix}}{3} = \begin{bmatrix} 2 & -5 & 3 \\ -5 & 12.7 & -8.7 \\ 3 & -8.7 & 12.7 \end{bmatrix}$, and the self-feature matrix of this step is $\mathcal{M} = \begin{bmatrix} 1 & -2 & -2 \end{bmatrix}^T \times \begin{bmatrix} 1 & -2 & -2 \end{bmatrix} = \begin{bmatrix} 1 & -2 & -2 \\ -2 & 4 & 4 \\ -2 & 4 & 4 \end{bmatrix}$

2) CONVOLUTIONAL ENCODER

We use a convolution encoder [38] to filter the noise in the data and encode the spatial patterns of feeding feature matrix sequence M. Four layers of fully convolution encoders are applied in our model to extract the features of the input feature matrix. Part (A) in Figure 2 is a four-layer convolutional encoding layer whose input is a sequence of feature matrices. We call the input of the first layer $O^{t,0}$ at this time t for convenience. To capture the tendency of the temporal sequence more accurately, we can take the feature matrix subsequence with more long-term information as the input of our model. At the time step t, the input is a length of w' feature matrix subsequences. The input $O^{t,0} = \left\{M^{t-(w'-1)\times s-1}, M^{t-(w'-2)\times s-1}, \cdots, M^{t-1}\right\}$ is a matrix of $w' \times k \times k$, where s is the stride to obtain the characteristic matrix and w' is the number of feature matrices. The output of l-th layer is given by:

$$O^{t,l} = f(W^l_{en} * O^{t,l-1} + b^l_{en}) \tag{3}$$

where W^l_{en} denotes the filter kernel of layer l, b^l_{en} is the bias, $*$ denotes the convolutional operation, $O^{t,l}$ is the output of layer l, $O^{t,l-1}$ is the output of the $l-1$ layer and meanwhile it is the input of layer l, and $f(\cdot)$ is an activation function.

3) CONVLSTM

The features extracted by the convolution encoder can only be obtained according to the input feature matrices. To obtain accurately predicted results, it is also necessary to obtain the temporal characteristics. The traditional LSTM is skilled in extracting important features of the sequence over time. Shi *et al.* [39] improved the LSTM model to the ConvLSTM model which has been developed to capture the temporal information in a video sequence. The output of each layer of the encoder is used as the input of ConvLSTM. As mentioned earlier, there are four convolutional layers of four ConvLSTM blocks, and each layer contains a ConvLSTM structure. The information of ConvLSM is shown in Figure 2 (B). At time step t, the output $O^{t,l}$ calculated by Eq. 3 in layer l of the convolutional encoder is used as one input of the l-th layer ConvLSTM, and the other input is the previous time $t-1$ hidden state $H^{t-1,l}$ in the hidden layer of layer l. The ConvLSTM cell [39] is formulated as:

$$\begin{aligned}
I^{t,l} &= \sigma\left(W^l_{oi} * O^{t,l} + W^l_{hi} * H^{t-1,l} + W^l_{ci} \circ C^{t-1,l} + b^l_i\right) \\
F^{t,l} &= \sigma\left(W^l_{of} * O^{t,l} + W^l_{hf} * H^{t-1,l} + W^l_{cf} \circ C^{t-1,l} + b^l_f\right) \\
C^{t,l} &= F^{t,l} \circ C^{t-1,l} + I^{t,l} \circ \tanh(W^l_{oc} * O^{t,l} + W^l_{hc} \circ H^{t-1,l} \\
&\quad + b^l_c) \\
P^{t,l} &= \sigma\left(W^l_{op} * O^{t,l} + W^l_{hp} * H^{t-1,l} + W^l_{cp} \circ C^{t-1,l} + b^l_p\right) \\
H^{t,l} &= P^{t,l} \circ \tanh\left(C^{t,l}\right)
\end{aligned} \tag{4}$$

where $I^{t,l}$ represents the output of the input gate, σ is the sigmoid function, W^l_{oi} is the filter kernel of the input gate, $O^{t,l}$ is the input of layer l, W^l_{hi} is the filter kernel of the input gate to process the input of the hidden layer at the previous time step, $H^{t-1,l}$ is the output of the hidden layer at the previous time step and it is also the input of the hidden layer at this time step, W^l_{ci} is the filter kernel of the cell state at the previous time step in the input gate, $C^{t-1,l}$ is the cell state at the previous time step, b^l_i is the bias of the input gate, $*$ denotes the convolution operation, $\circ$ represents the Hadamard product, $F^{t,l}$ shows the output of the forget gate, $C^{t,l}$ is the updated state of the cell at time t, $P^{t,l}$ denotes the output at time t, $H^{t,l}$ is the state of the hidden layer at time t, and tanh is another activation function. When $t = 1$, there is no hidden layer state and cell state at time $t = 0$, and we set $H^{0,l}$ and $C^{0,l}$ to zero matrices of the same size as the input. These parameters $I^{t,l}, F^{t,l}C^{t,l}, P^{t,l}, H^{t,l}$ and $O^{t,l}$ have the same size.

4) CONVOLUTIONAL DECODER

The opposite of coding layer is decoded one. We can reconstruct the input data into the extracted feature output. The decoding layer reconstructs a sub-feature matrix sequence from the input $O^{t,0}$ into a sub-feature matrix sequence $\hat{O}^{t,0}$ of the same size as the input, and then we can predict the self-feature matrix $\hat{\mathcal{M}}^t$ that only contains its own information according to the decoded characteristic matrix subsequence $\hat{O}^{t,0}$ at the time t. We follow the idea mentioned in [34] and reversely decode the deconvolution from layer $l = 4$ to layer $l = 1$ to reconstruct the matrix of each layer. If in the last layer $l = 4$, we directly deconvolve the output $P^{t,4}$ of the ConvLSTM at the last layer $l = 4$ and reconstruct the matrix of the previous layer $\hat{O}^{t,3}$. If it is not in the last layer $0 < l < 4$, the step is to combine the output $P^{t,l}$ of the CovnLSTM in layer l calculated by Eq. 4 with the output $\hat{O}^{t,l}$ of the deconvolution layer $l+1$ calculated by Eq. 5. Then the deconvolution operation is performed in the result of concatenation to reconstruct the feature matrix sequence of the $l-1$-th layer. The concatenation and deconvolution operations are shown in Figure 2 (C), and the output of (C) is a reconstructed feature matrix. The convolutional decoder which is formulated as:

$$\hat{O}^{t,l-1} = \begin{cases} f\left(W^{t,l}_{de} \circledast P^{t,l} + b^{t,l}_{de}\right), & l = 4 \\ f\left(W^{t,l}_{de} \circledast \left[P^{t,l} \oplus \hat{O}^{t,l}\right] + b^{t,l}_{de}\right), & 0 < l < 4 \end{cases} \tag{5}$$

where $\circledast$ denotes the deconvolution operation, $\oplus$ is the concatenation operation, $f(\cdot)$ is the activation unit (same as the encoder), and $W^{t,l}_{de}$ and $b^{t,l}_{de}$ are the filter kernel and bias parameter of l-th deconvolutional layer. The output $\hat{O}^{t,0}$ of the final convolutional decoder (with the same size of the input matrices) denotes the representations of reconstructed feature matrices. The output $\hat{O}^{t,0} = \left\{\hat{M}^{t-(w'-1)\times s-1}, \hat{M}^{t-(w'-2)\times s-1}, \cdots, \hat{M}^{t-s-1}, \hat{M}^{t-1}\right\}$ can be regared as a matrix with a size of $w' \times k \times k$. The decoder is able to incorporate feature maps at different deconvolution and ConvLSTM layers, which is effective in improving anomaly detection performance.

A linear structure is added after the last $l = 1$ layer of decoding to predict the self-feature map $\hat{\mathcal{M}}^t$ at time t. The output $\hat{O}^{t,0}$ of the decoder at the $l = 1$ layer is taken as the input of this linear structure, and the output $\hat{\mathcal{M}}^t$ of this linear structure is the predicted self-featrue matrix with the same size as $\mathcal{M}^t$ of $k \times k$. As shown in Figure 2, the input of (D) is the reconstructed feature matrix sequence decoded from the output of (C), and the output of (D) is its predicted self-feature matrix predicted conforming to the current temporal trend. We set the formulation of the linear structure in the last layer as:

$$\hat{\mathcal{M}}^t = W^t \times \hat{O}^{t,0} + b^t \tag{6}$$

5) LOSS FUNCTION

There are three kinds of the commonly used loss functions in regression tasks: Mean Squared Loss (MSE), Root Mean Squared Loss (RMSE), and Mean Absolute Loss (MAE), where RMSE is the square root of MSE. Willmott and Matsuura [40] introduced the respective advantages of MAE and RMSE loss functions. When the error is very large, the result calculated by MSE will be much larger than the result of MAE, and the gradient of MSE will also change with the size of the error due to the square relationship of MSE. Therefore, MSE loss is a loss function that is more suitable to the field of anomaly detection. Our loss function is formulated as follows:

$$LossFunc = \frac{\sum_{\mu=1}^{k}\sum_{\nu=1}^{k}\left(\mathcal{M}_{\mu\nu}^t - \hat{\mathcal{M}}_{\mu\nu}^t\right)^2}{k \times k} \tag{7}$$

where $\mathcal{M}^t$ is the self-feature matrix at time step t, $\hat{\mathcal{M}}^t$ is the predicted the self-feature matrix at time step t, $\mu\nu$ is the position of the μ-th row and ν-th column in the matrix, and the size of both kinds of matrices is $k \times k$.

D. THRESHOLD-SETTING STRATEGY

According to the definition in section III-B2, there is a certain value that indicates the extent to which the current value deviates from the historical data. The anomaly score calculated by the proposed method represents a certain mapping value in the definition. Thus, we also need to set a threshold of anomaly scores to judge whether there is an abnormal event. The setting of the threshold is a key task in anomaly detection [41]. In our work, the data used in the training model are normal data, and the abnormal scores calculated with these training data also belong in the normal range. In this paper, we propose a method to set the threshold according to the training set. This can provide a valuable reference for setting the abnormal boundary. Considering that noise and individual extreme data in the training set can affect the threshold setting, instead of directly using the maximum of the training set's anomaly score as the result of the threshold setting, the anomaly scores of the training set are analyzed again. Algorithm 2 displays the strategy of threshold setting. The threshold-setting algorithm first calculates the difference matrix sequence S according to the real self-feature matrix sequence $\mathcal{M}$ and the predicted self-feature matrix sequence $\hat{\mathcal{M}}$ computed by Eq. 6. Then, the upper quartile of the set S' consisting of the maximum value of each difference matrix is calculated to obtain the threshold value θ that can contribute to the abnormal score. According to this threshold θ, the number of the difference matrix sequences S at each time step of training set exceeding threshold θ is recorded as the anomaly score $S_{anomaly}$. Then, we calculate the upper quartile of the abnormal score $S_{anomaly}$ of the training set and record it as the temporary threshold δ'. Finally, through the relationship between the abnormal score $S_{anomaly}$ and the temporary threshold δ', the threshold δ of the abnormal score is obtained according to the situation of the score. If there is no abnormal score greater than the temporary threshold δ', the threshold δ takes the maximum value of the abnormal score. If there is a value in $S_{anomaly}$ greater than the temporary threshold, the value of δ is the upper quartile of the part where the abnormal score is greater than the threshold δ'. This threshold setting strategy takes both the noise of the data and the extreme data into account and makes a secondary analysis of anomaly scores of normal datasets, so the threshold value obtained is also relatively reasonable.

The detailed process of the threshold setting strategy is as follows. First, we calculate the difference matrix S according to the self-feature matrix $\mathcal{M}$ and the predicted self-feature matrix $\hat{\mathcal{M}}$ at every time step. S^t with size $k \times k$ is the result of squaring the value of subtracting $\mathcal{M}^t$ from $\mathcal{M}^t$. Our formula for S^t is as follows:

$$S^t = \left\|\mathcal{M}^t - \hat{\mathcal{M}}^t\right\|^2 \tag{8}$$

According to Eq. 8 we can calculate the difference matrices at all time steps and obtain a series of difference matrices named S with the same length as the training data, where S^t is the t-th matrix in S. The calculation of the anomaly score at the time t needs the elements in S^t, so we set another threshold θ to determine whether the elements in S^t can contribute to the anomaly score. The setting of θ is also an essential operation. The maximum value $max\left\{S^t\right\}$ is taken from the difference matrix at each time step to form a set S' whose size equals to the length of training data. The value of θ is set according to the statistics-based method presented in [42], where the abnormal part of the data is detected by the interquartile distance (IQR). IQR is defined as follows:

$$IQR = Q_3 - Q_1 \tag{9}$$

where Q_3 is upper quartile and Q_1 is lower quartile. Then, the normal range of data is calculated as:

$$upper - bound = Q_3 + IQR \times factor \tag{10}$$

$$lower - bound = Q_1 - IQR \times factor \tag{11}$$

where $factor$ is usually set to 1.5 or can be set to the appropriate value as needed. We only need to calculate the upper limitation of date because anything below the lower limitation calculated by Eq. 11 is a more normal value. We set $factor$

to 1.5 as usual, and according to Eq. 9 and Eq. 10 the θ is calculated as follows:

$$\theta = 1.5 \times \left(Q_3\left(S'\right) - Q_1\left(S'\right)\right) + Q_3\left(S'\right) \tag{12}$$

where S' is the maximum set of matrices in the difference matrix S of the training set, $Q_3\left(S'\right)$ is the upper quartile of S', and $Q_1\left(S'\right)$ is the lower quartile of S'.

We have obtained the threshold θ from Eq. 12 to determine whether the elements in the difference matrix S^t can contribute to the abnormal score at any time t. Then, we can calculate the abnormal score at this step. The number of S^t that is greater than threshold θ at time t is counted as the abnormal score at this time, and the abnormal score at t is calculated as follows:

$$S^t_{anomaly} = F\left(\left\|\mathcal{M}^t - \hat{\mathcal{M}}^t\right\|^2\right) \tag{13}$$

$F(\cdot)$ is the function to count the number of elements in the difference matrix S^t greater than θ, and $\hat{\mathcal{M}}^t$ and $\mathcal{M}^t$ denote the predicted self-feature matrix and real self-feature matrix, respectively, at time t. We can calculate the abnormal score at any time in the dataset according to Eq. 13, and the abnormal score in the training set is recorded as *STrain*. In this case, we need to calculate the boundary of the abnormal score δ to judge whether this score of time t can determine the observed value as anormaly.

Since all the data in the training set contain noise, we do not apply the maximum abnormal score in the training set as the value of the abnormal boundary δ in order to avoid the setting of the threshold being influenced by individual data with significant noise. Instead, we first use *STrain* to calculate a temporary upper bound δ' according to the abnormal situation in the training set. Eq. 9 and Eq. 10 are still used to calculate the temporary upper-boundary, $\delta' = 1.5 \times (Q_3\,(STrain) - Q_1\,(STrain)) + Q_3\,(STrain)$, where $Q_3\,(STrain)$, $Q_1\,(STrain)$ are the upper and lower quartiles of *STrain*, respectively. The setting of of value δ is divided into two cases. If all of the anomaly scores of the training set are less than the temporary upper bound δ', then the anomaly boundary δ takes the maximum of *STrain*. If the anomaly score of the training set has a value greater than the temporary boundary, then the boundary δ takes the upper quartile of the portion of *STrain* that exceeds the temporary upper limitation. We calculate δ by the following formula:

$$\delta = \begin{cases} max\left\{STrain\right\}, & \forall\beta \in S_{anomaly} \leq \delta' \\ Q_3\left(\left\{\beta | STrain > \delta'\right\}\right), & others \end{cases} \tag{14}$$

After the threshold is set, the testing set can be inputted into the trained model, and the abnormal score of t at a certain time step is obtained. If the abnormal score is greater than δ, then this time step is considered to be abnormal and far from the expected behavior. Significantly, the method of setting the threshold is only a reference to the data, and the specific threshold setting needs to be adjusted according to the different scenarios.

Algorithm 2 Threshold Setting Algorithm

Input: real self-feature matrix sequence $\mathcal{M}$ of training set, real self-feature matrix sequence $\hat{\mathcal{M}}$ of training set, length of matrix sequence n

Output: threshold of statistical anomaly score θ and threshold of determining anomaly δ

for int $t = 1$ **to** n **do**
  $S[t] \leftarrow \left\|\mathcal{M}[t] - \hat{\mathcal{M}}[t]\right\|^2$ ▷ Calculate the t-th difference matrix
  $S'[t] \leftarrow 0$
  for int $i = 1$ **to** k **do**
    for int $j = 1$ **to** k **do**
      if $S'[t] < S[t][i][j]$ **then**
        $S'[t] \leftarrow S[t][i][j]$ ▷ Select the maximum value in t-th the difference matrix
$\theta \leftarrow upper - bound(S')$ ▷ Apply upper-bound to the maximum set in the difference matrix Eq. 10
for int $t = 1$ **to** n **do**
  $S_{anomaly}[t] \leftarrow 0$
  for int $i = 1$ **to** k **do**
    for int $j = 1$ **to** k **do**
      if $S[t][i][j] > \theta$ **then**
        $S_{anomaly}[t] \leftarrow S_{anomaly}[t] + 1$
$\delta' \leftarrow upper - bound(S_{anomaly})$ ▷ Apply upper-bound to the anomaly score set of training set Eq. 10
if $\forall\beta \in S_{anomaly} < \delta'$ **then**
  $\delta = max\left\{S_{anomaly}\right\}$
else
  $\delta = Q_3(S_{anomaly} > \delta')$ ▷ Apply the upper quartile to the anomaly score set of all $S_{anomaly}$ greater than δ'
return θ, δ

IV. EXPERIMENT AND DISCUSSION

In this section, we use synthetic data and real data respectively to test the detection effect of our proposed model MCRAAD and test the influence of some other factors on our model with several groups of synthetic datasets. In the experiments with the artificial data, we first introduce a method of generating the simulated data, and then design a group of comparison experiments between these basic models and the MCRAAD model on the anomaly detection effect with the synthetic data. To test the robustness of the proposed model, we set up a group of comparison experiments with different noise sizes. The influence of input data setting and network structure on MCRAAD is also shown in this section. Finally, we compare the detection performance of several basic models and our model on real dataset monitoring home information.

In our experiments, we compare MCRAAD with seven baseline methods, i.e., One-Class SVM (OCSVM), Isolation Forest (iForest), History Average (HA), Auto Regression Moving Average (ARMA) [43], Long Short-Term Memory (LSTM), Autoencoder, and LSTM encoder-decoder (LSTM-ED). OCSVM and iForest output the anomaly label

directly, and other models used for comparison directly use the mean square error loss as the anomaly score. We follow the method of training data used in OCSVM, which only trains one kind of data. The input of the proposed model is normal data without labels and the labels of the dataset are used only to evaluate the performance of the model.

Several metrics need to be set to evaluate the results of the experiments. In unbalanced data, to evaluate the detection performance of these models more reasonably, four metrics, i.e., Precision, Recall, F1 Score, and Gmean, are used to evaluate the anomaly detection performance of each method. The geometric mean is also an extensively used metric in imbalanced data scenarios [44]. It is the geometric mean between recall and specificity, and is computed as Eq. 15:

$$Gmean = \sqrt{\frac{TP}{TP+FN} \times \frac{TN}{TN+FP}} \tag{15}$$

These metrics are calculated from a confusion matrix, which displays the crossing correct and wrong predictions between pairs of categories. It represents True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN) undertaken by the system.

A. EXPERIMENTS ON SYNTHETIC DATA

We use easily controlled periodic functions to generate simulated data. Each time-series is formulated as:

$$Data\,(t) = \begin{cases} \overbrace{sin}^{c1}\overbrace{[a \times t + b]}^{c2} + \overbrace{\lambda\epsilon}^{c3}, & s_{rand} = 0 \\ \overbrace{cos}^{c1}\overbrace{[a \times t + b]}^{c2} + \overbrace{\lambda\epsilon}^{c3}, & s_{rand} = 1 \end{cases} \tag{16}$$

where c1 is a trigonometric function that can simulate temporal patterns. These two trigonometric functions are common trigonometric functions with good periodicity and strong regularity, and their ranges are $[-1, 1]$. Part $c2$ can control the periodicity and starting position of the trigonometric function. To make the data pattern of each dimension not identical, a and b in $c2$ are set as random numbers. The larger the value of a is, the smaller the period is. c3 simulates data noise as well as various shapes. The value of ϵ is a random number that meets the standard normal distribution $N(0, 1)$, and λ is a parameter to control the size of the noise; its range is $[0.1, 0.5]$. S_{rand} is a random number with values of 0 and 1 to set the trigonometric function used to generate the data of the current dimension. The sine function is used for 0, and the cosine function is used for 1. In addition, we also need to add abnormal waves as anomalies by randomly selecting the frequency and phase of each time-series.

1) SYNTHETIC DATA

The synthetic time-series data can be generated by the Eq. 16. We set the parameter of noise λ to 0.15 and randomly generate 20 time-series, each including 12000 points. The first 5000 of these points are used as the training set are normal data with noise. The testing set including the last 7000 points contains four randomly added anomalies with different sizes and lengths (4 is a random integer between 2 and 5). The method of adding abnormal fluctuations is to add a random number $[0, 1]$ to the starting position of the abnormal fluctuation after standardizing the dataset and set the successive length to the same value as the starting position of the anomaly. We normalize the multivariate time series data by the z-score normalization method with a mean of zero and a standard deviation of one.

When using Algorithm 1 to calculate the feature matrix at each time step, the window size w is set as 10. When taking the training set and the testing set from the feature matrix sequence, we set the step size s to 5. We input 10 feature matrices ($w' = 10$) to the model at one time. These parameters are relatively better parameters found by network search algorithm. We follow the network structure and parameters in [34] as the initial setting of our network architecture and parameters and make some changes to adapt to our experiments. The four-layer convolution kernels in these convolutional encoders are set to 32 kernels of size $10 \times 3 \times 3$ with 1 stride, 64 kernels of size $32 \times 3 \times 3$ with 2 strides, 128 kernels of size $64 \times 2 \times 2$ of with 2 strides, 256 kernels of size $128 \times 2 \times 2$ with 2 strides. The kernel setting of ConvLSTM is related to the encoding layer. We maintain the same size of convolutional kernel as the convolutional encoder at each layer and the stride is set to 1. In addition, the padding is set to an appropriate value to keep the size of the input and output of ConvLSTM consistent. The settings of the convolutional decoder from the fourth layer to the first layer are 128 kernels of size $256 \times 2 \times 2$ with 2 strides, 64 kernels of size $256 \times 2 \times 2$ with 2 strides, 32 kernels of size $128 \times 3 \times 3$ with 2 strides, 10 kernels of size $64 \times 3 \times 3$ with 1 stride. It is worth noting that the output of each layer of the decoder must have the same size as the input size of the encoder. We use the Scaled Exponential Linear Unit (SELU) [45] as the activation function of four encoders and decoders. The activation function SELU has a self-normalization feature that can converge to a mean of 0, a variance of 1, or a variance with upper and lower bounds even when noise is added. The parameters of the activation function used in the experiments are the default parameters. We also take the advantage of the Adam optimizer [46] in training. The loss function can be found in Eq. 7. In the experiments, we take advantage of the grid search algorithm to select the hyperparameters for these models in advance. The three deep learning models, Autoencoder, LSTM, and LSTM-ED adopt the uniform distribution initialization model and their activation functions are ReLU and the optimizer are Adam. The proposed model adopts the standard normal distribution initialization mode. In the training process, if the performance of network structure is not improved after two epochs, the batch size would be halved until the batch size was equal to 32, and the learning rate would be reduced to half of the original. If the network still cannot be improved after the learning rate and batch size were reduced, the training operation would be stopped. When training our model with the synthetic time-series, we let epoch equal to 15 and the

TABLE 2. Anomaly detection results of eight models on synthetic data.

Model	Precision	Recall	F1 Score	Gmean
OCSVM	0.386	0.309	0.343	0.552
iForest	0.330	0.377	0.352	0.608
HA	0.670	0.337	0.449	0.624
ARMA	0.723	0.380	0.498	0.615
LSTM	0.462	0.380	0.417	0.613
Autoencoder	0.639	0.475	0.545	0.687
LSTM-ED	0.929	0.726	0.815	0.852
MCRAAD	**0.986**	**0.806**	**0.887**	**0.897**

* The best values in each metric are highlighted in bold.

batch equal to 64. After training, we use the trained model to test the training set and obtain the threshold of anomaly score based on the predicted training set and the real training set according to Algorithm 2. When calculating the feature matrix, we set the feature matrix of the data that does not meet the conditions to matrix zero. Thus, this part of the data does not participate in the training and testing process. In addition, In order that the feature matrices input to the model do not contain zero matrices, we start training and predicting from the 55th iteration, and the data used in calculating the loss and anomaly score and evaluating the model do not include the first 55 datasets.

We repeat these experiments on the same simulated dataset with eight models, and the average results are shown in Table 2. The F1 score below 0.4 for the two machine-learning approaches illustrate that temporal prediction models can better capture the developmental characteristics of the data. The HA and ARMA models pay more attention to precision but fail to find more true abnormal positions, and the values of their precision are near twice the recall values. Among these basic models, the LSTM-ED model, which adapts to dealing with noise and can capture time characteristics, performs better than the other six basic models, and its indicators are only secondary to MCRAAD. MCRAAD performs best on all models in terms of precision and recall score from the detection results of the eight models. The improvements over the best baseline range from 5% to 11%. The results calculated by the seven basic models have more false negatives and false positives than our method. From the perspective of the comprehensive F1 score and Gmean score, our model has the best anomaly detection performance among these models.

To display the results of these models more directly, we draw a group of line charts with the abnormal scores of these models. Since OCSVM and iForest directly give the abnormal results without abnormal scores, we only plot the anomaly scores of the six prediction models. The result is shown in Figure 3. The abscissas in the figure are the time marks of the time-series data with a length of 7000, the ordinate denotes the abnormal score, and the dashed green

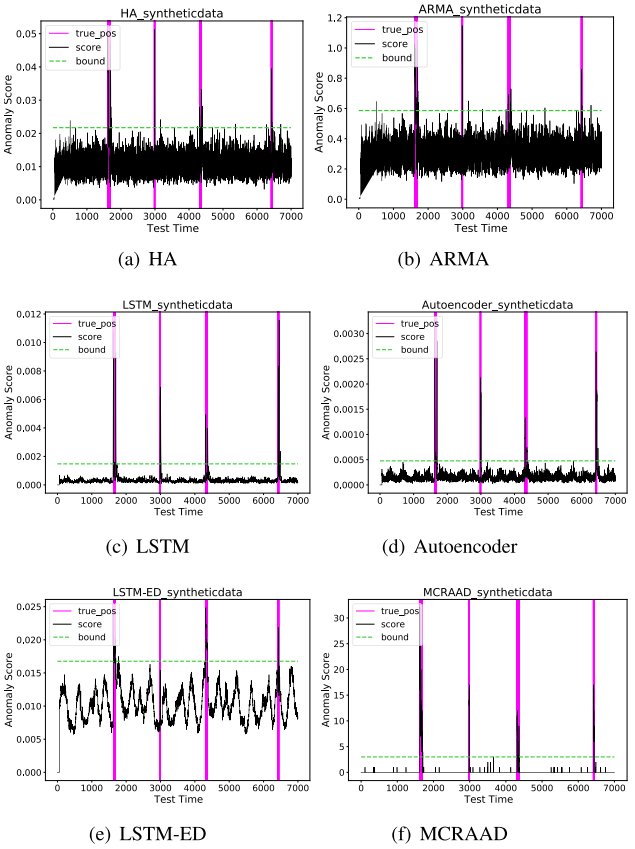
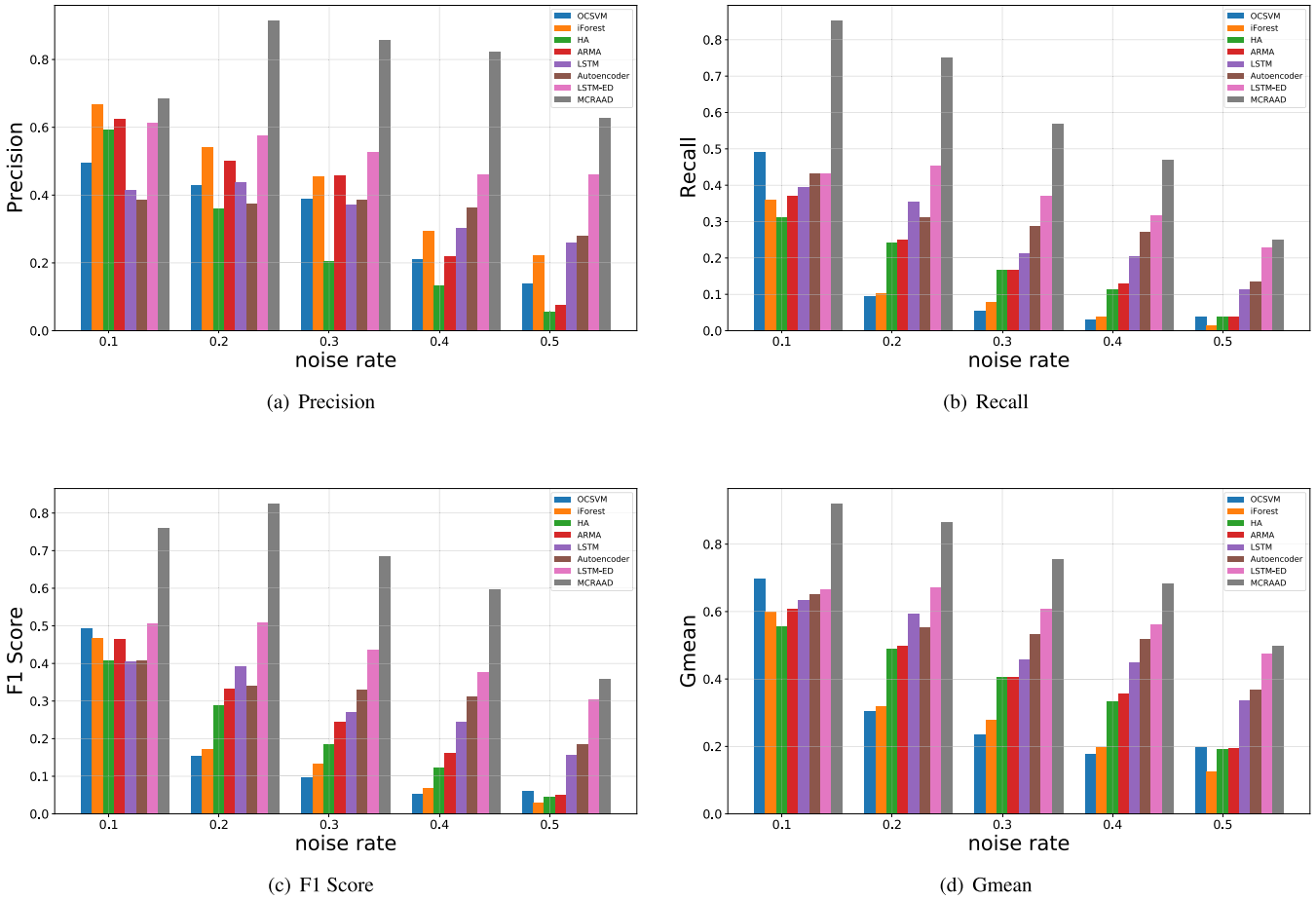


FIGURE 3. Results of anomaly detection with synthetic data by six prediction models. In each subgraph, abscissa denotes time steps, and ordinate is abnormal score. Black line in graph is abnormal score line chart drawn, and green dotted line is boundary dividing abnormal and nonabnormal. Upper part of green line is abnormal, and lower part is normal. Title of the subgraph is name of prediction model used. (a)(b) are results of statistics-based prediction approach, (c)(d)(e) are results of basic deep-learning prediction model, and (f) is result of proposed MCRAAD model.

line is the boundary dividing whether it is abnormal. The upper part of this line is abnormal points, while the lower part is normal. The magenta vertical lines mark the true abnormal positions. The data have four abnormal fluctuations, and each abnormal fluctuation contains a different number of abnormal points. The four subgraphs (a)-(d) only determine a small number of abnormal points in the abnormal fluctuations, and the detected abnormal points contain more normal points. Although more abnormal points can be accurately detected in (e), only three abnormal fluctuations can be detected. By contrast, MCRAAD in (f) can detect all abnormal fluctuations and accurately identify more abnormal points in these fluctuations.

2) NOISE

To check the robustness of the model MCRAAD, we use Eq. 16 to generate five sets of simulated data with a length of 10000 and a dimension of 10. The parameters λ of five sets of data are 0.1, 0.2, 0.3, 0.4, 0.5, respectively. The first 4000 points are used as the training set and the last



(a) Precision (b) Recall

(c) F1 Score (d) Gmean

FIGURE 4. Performance pictures of eight models' adaptability to noise. (a)(b)(c)(d) are precision, recall, F1 score, and Gmean of eight models, respectively, whose value ranges are [0, 1]. Abscissa of each subgraph is severity of added noise. There are five levels of noise: 0.1, 0.2, 0.3, 0.4, and 0.5. From left to right of abscissa, numbers represent increasingly severe noise, and each color of histogram represents a model.

6000 points with abnormal fluctuations of different sizes at the same position are used as the testing set.

These eight models are used to detect the five sets of data with different noises. We plot the results into four histograms, as shown in the Figure 4. The ordinates of (a)(b)(c)(d) in Figure 4 are the precision, recall, F1 score, and Gmean of these models, and the abscissas of these figures from left to right represent increasing noise. As the noise becomes larger and larger, the comprehensive detection ability of these models shows a downward trend. When the noise level is 0.1, our model does not perform well with regard to precision; this result is close to that of other models. The detection performance of OCSVM and iForest when the noise is equal to 0.1 is slightly better than that of the common prediction models HA and ARMA. However, the detection performance of these two machine learning models decreases sharply due to the more serious noise, and they can only focus on precision and not recall. As noise becomes increasingly serious, models that can capture historical temporal information are becoming increasingly prominent. In particular, the precision of MCRAAD is no less than 0.1 to 0.3 higher than that of the other models. From the perspective of recall, our model has been in the best state. From the perspective of comprehensive analysis, MCRAAD is more adaptive to noise and has better robustness.

3) IMPACT OF THE INPUT DATA

The detection performance of MCRAAD is also affected by the input data in multivariate time series. If the input data contains little historical data, the model is more susceptible to noise. while longer-term data contains more obsolete patterns, which also degrades the detection performance of the multidimensional time series anomaly detection model. In the proposed framework, the input of the model is jointly controlled by the window size, step size, and the number of feature matrices input to the model. The three aspects are evaluated by the control variable method, and the results are shown in Figure 5. Figure 5 (a) shows that the window size has little effect on the accuracy of MCRAAD, but a significant impact on the recall capability. When the minimum window is set to 4, the recall index of the model is only 0.4. As the window size increases, so does the performance of the model. However, when the size of the window exceeds the appropriate value, the detection performance of the model

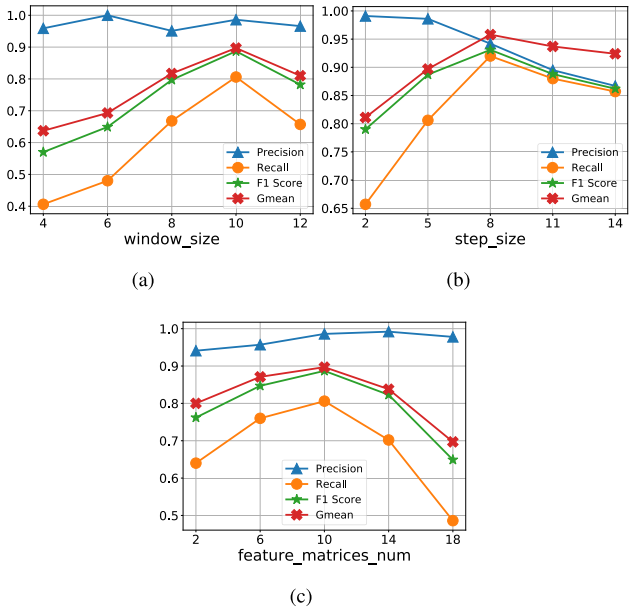


FIGURE 5. The performance results of different input settings. The ordinates of all subgraphs represent the values of the four metrics. The blue line represents the precision, the orange line represents the recall, the green line represents F1 score, and the red line represents Gmean. The abscissas of (a)(b)(c) represent the window size, the step size of the data, and the number of feature matrices input into the model respectively.

tends to decline. When the window is small, a feature matrix contains little information and is more susceptible to noise. When the window is too large, a feature matrix contains longer-term information that can mislead the model. It can be seen from the above that window size can affect the detection performance of the model, and the improper size of the window can reduce the performance of the model. Figure 5 (b) and (C) show the effect of step size and the number of feature matrices input into the model at one time on the detection performance of the proposed framework respectively. The experimental results of both are similar to Figure (a). Similarly, values that are too large or too small can reduce the performance score of the model. If these three parameters can get the most appropriate value, it will greatly improve the detection performance of the model. Therefore, in the experiments, we set several groups of values for each parameter respectively, and use the network search algorithm to extract relatively appropriate values to achieve better detection results.

4) TIME PERFORMANCE

In this section, we will discuss the time performance of each model. In our experiments, we use a 960M and an i7-4720HQ CPU with 16G of memory. the training and testing times for these models are shown in Table 3. From Table 3, we can obtain that the statistical-based HA model without training time uses the mean value of historical data as its predictive value directly. In our experiments, the ARMA model trains historical data once for each prediction, so the training time of the ARMA model is much longer than the traditional machine learning models and other statistical-based models in the paper. The training time and testing time of both models based on traditional machine learning are less than one second. Deep learning-based anomaly detection models take longer than other models. The training time of MCRAAD is more than ten times that of the LSTM-ED model and the testing time is nearly five times that of the LSTM-ED model. Although MCRAAD takes much time in training and testing, its detection performance has been superior to these evaluated models in these experiments. With the development of hardware devices, the impact of time-consuming problems will become smaller and smaller.

TABLE 3. Time performance of the eight models.

Model	Train time(s)	Test time(s)
OCSVM	0.2	0.1
iForest	0.9	0.7
HA	/	0.2
ARMA	124.5	13.3
LSTM	243.4	38.2
Autoencoder	131.2	9.7
LSTM-ED	354.8	42.1
MCRAAD	3829.3	205.7

* '/' means that the model does not need to spend time at this stage.

TABLE 4. Anomaly detection results of MCRAAD with different settings.

Model	Precision	Recall	F1 Score	Gmean
MCRAAD$^{(1)}$	0.375	0.017	0.033	0.131
MCRAAD$^{(1,2)}$	0.447	0.120	0.232	0.346
MCRAAD$^{(4)}$	0.820	0.417	0.553	0.645
MCRAAD$^{(3,4)}$	0.978	0.766	0.859	0.875
MCRAAD	**0.986**	**0.806**	**0.887**	**0.897**

* The best values in each metric are highlighted in bold.

5) SETTING OF THE FRAMEWORK

To demonstrate the influence of network structure and parameters on detection performance, we conduct a group of comparative experiments. The other four settings are MCRAAD$^{(1)}$ with only the first layer network structure, MCRAAD$^{(1,2)}$ with the first two layers network, MCRAAD$^{(4)}$ with only the fourth layer network, and MCRAAD$^{(3,4)}$ with the last two layers network. The experimental results are shown in Table 4. MCRAAD$^{(1)}$ with only one layer of models has the worst results with the recall equals to 0.017. The recall of the MCRAAD$^{(1,2)}$ is nearly ten times higher than that of the MCRAAD$^{(1)}$. The same trend is also present in MCRAAD$^{(4)}$ and MCRAAD$^{(3,4)}$. The number

of convolution kernels with the same number of layers also affects the detection results. The recall and F1 Score of MCRAAD$^{(4)}$ with 256 kernels are much higher than that of MCRAAD$^{(1)}$ with only 32 convolution cores. Its precision is 2.6 times that of MCRAAD$^{(1)}$, and its Gmean is 6.7 times that of MCRAAD$^{(1)}$. It can be concluded from the above results that the number of layers and convolution cores of the model can affect the feature extraction and detection performance of the model. Without a sufficient number of layers and kernels, the framework cannot extract more information that is more useful for anomaly detection. The results of MCRAAD with a four-layer network structure and multiple kernels are the best among this group of experiments, which is also relatively consistent with the fact.

TABLE 5. Anomaly detection results of eight models on real data.

Model	Precision	Recall	F1 Score	Gmean
OCSVM	0.246	0.299	0.270	0.543
iForest	0.217	0.243	0.229	0.490
HA	0.344	0.288	0.314	0.535
ARMA	0.476	0.351	0.404	0.591
LSTM	0.652	0.405	0.501	0.636
Autoencoder	0.678	0.360	0.471	0.601
LSTM-ED	0.780	0.432	0.583	0.657
MCRAAD	**0.880**	**0.825**	**0.852**	**0.908**

* The best values in each metric are highlighted in bold.

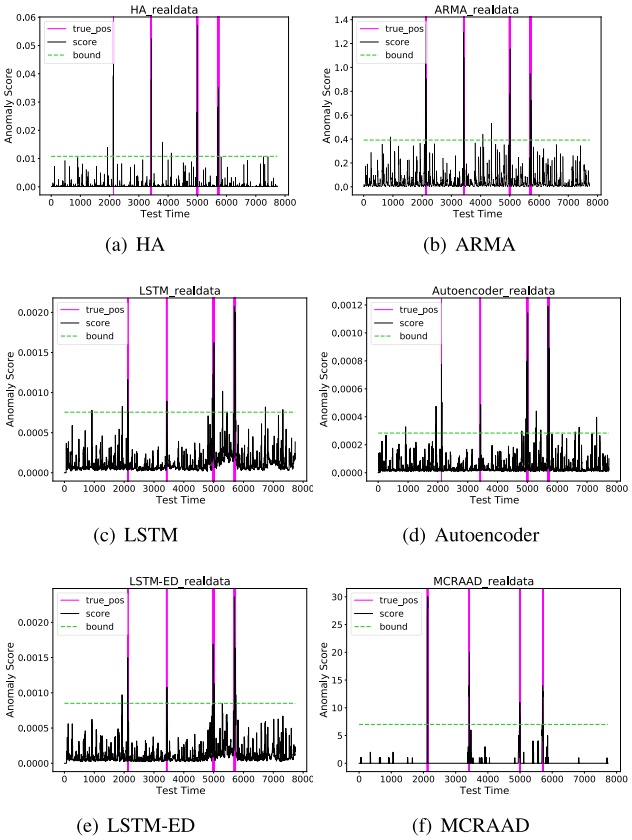
B. EXPERIMENTS ON REAL DATA

MCRAAD can be used to detect anomalies in the real world; for example, our work can establish a housing emergency warning system that can raise alarms for emergencies such as fires and reduce losses as much as possible. This alarming system is displayed in Figure 1. The sensor data for monitoring house indicators can be regarded as multivariate time-series data, as shown in (C). This box contains multiple sensors for monitoring house attributes and the data generated by these sensors are multivariate time-series data shown in (A). The distinctive colors represent different series. To test the detection ability of our proposed MCRAAD method in the real world, we use a time-series dataset that records the temperature and humidity of different rooms in a house in a real environment. The temperature and humidity of the room are monitored through the ZigBee wireless sensor network. Each node of the sensor records the transmitted temperature and humidity every 3.3 minutes, and records the data on average every ten minutes. The weather data in the dataset are downloaded from the nearest airport weather station and merged with the recorded data. The data contains a total of 29 attributes. We remove irrelevant attributes and finally retain only 20 attributes such as temperature and humidity. The total length of the data is 19735. We randomly select a time step from the first 6000 points as the starting step of intercepting the data, the data from the selected point to the 12000th time step as the training set, and all the data after 12000 as the testing set. We randomly select dimensions and locations to add four abnormal fluctuations as simulated emergencies (similar to what we did in the synthetic data). The results of multiple experiments with the eight models on the real dataset are shown in Table 5.

Table 5 shows that the unpredictable noise in practical works has a great influence on the detection effect of the HA and ARMA models. All of these indicators of iForest are the lowest compared with other models. Data in the real world do not have a fixed probability distribution, and the iForest model randomly selecting features to divide the range and detecting anomalies has the worst performance among all models in the real data. However, the F1 score of this classification model OCSVM is 0.270, slightly higher than that of iForest, and this model can only make anomaly classification according to the characteristics of each dataset and cannot use the relationship between time-series data to assist the anomaly judgment, which also leads to its poor detection performance in the real dataset. The autoencoder, which can cope with noise but cannot capture the temporal characteristics well, has a higher detection precision of 0.678 but a lower ability to search for all anomalies.

The LSTM model with an F1 score of 0.501 and Gmean score of 0.636, which can capture temporal features but is easily affected by noise has better comprehensive detection capabilities than these models, which cannot capture the time pattern. Among the basic models, the LSTM-ED, which can not only capture the time-series characteristics well but also reduces the influence of noise, has the strongest detection ability. Compared with other models, MCRAAD has the highest value of all metrics among these eight models and the best detection capability in these evaluated models. As shown in Figure 6, we also plot the abnormal scores detected by the six prediction models on the real data. (a)(b) are the anomaly score line charts of model HA and model ARMA. (c)(d)(e) are the results of three deep learning models. We find that these basic models cannot detect anomalies in real data accurately and comprehensively. Relatively speaking, our model performs better in the anomaly detection of real data from subfigure (f) marking the anomaly boundary and the real abnormal points. Thus, in some real application scenarios, the anomaly detection performance of the MCRAAD model is slightly better than the comparison of several basic models.

According to the results of the all the above comparative experiments, our method is more suitable for the role of abnormal detection in a home emergency alarm system as shown in Figure 1. Based on historical house monitoring data recorded by (C), MCRAAD simulates the relationship between time-series data at every time step and its historical data as much as possible. MCRAAD can predict the data at the next step in advance based on the historical data input to the model. When the new data collected by



(a) HA (b) ARMA

(c) LSTM (d) Autoencoder

(e) LSTM-ED (f) MCRAAD

FIGURE 6. Results of anomaly detection with real dataset by six prediction models. In each subgraph, abscissa denotes time steps, and ordinate is abnormal score. Black line in graph is abnormal score line chart drawn, and green dotted line is boundary dividing abnormal and nonabnormal. Upper part of green line is abnormal, and lower part is normal. Title of the subgraph is name of prediction model used. (a)(b) are results of statistics-based prediction approach, (c)(d)(e) are results of basic deep-learning prediction model, and (f) is result of proposed MCRAAD model.

the sensor arrive, we immediately compare the difference between the real data and the predicted data to obtain the anomaly score at this time step, and use the threshold that has been set to determine whether the anomaly score reaches the anomaly category. If the anomaly score exceeds the threshold, the connected alarm issues an alarm immediately to notify people that an emergency may have occurred in the house.

C. DISCUSSION

In this part, we analyze and summarize the advantages of MCRAAD. The performance of MCRAAD is influenced by many factors, such as the window size, the number of input feature matrices, the step size, the number of kernels, etc. The results of the experiments show that the proposed method on both synthetic datasets and real datasets is slightly better than the other seven evaluated algorithms. The reasons why MCRAAD is superior to the basic model of comparison can be summarized as follows: First, when preprocessing data, a feature matrix is calculated for each time step, which contains the relationship between its own information and the information of a sub-sequence. In addition, the process of calculating the feature matrix is also a process of amplifying features and reducing noise. The feature matrix provides the first guarantee for MCRAAD to extract the feature and deal with the noise. Second, a stacked autoencoder structure with ConvLSTM cells is applied to extract the features of multivariate time series. In the decoding phase, the information of each hidden layer is used again to automatically extract deeper and more representative features of the data, which is also the process of reducing the impact of noise. Predictions made from reconstructed data with important characteristics are also data that corresponds relatively to data trends. Third, only the normal dataset can be used as the training set when training the model. Therefore, the trained model can be well reconstructed and predict the normal data. The results reconstructed for abnormal data models will differ greatly from the real results. A strategy of threshold setting with normal training sets is applied in this model. This strategy takes noise and extreme data in normal datasets into account and makes a secondary analysis of anomaly scores in normal datasets. Therefore, the threshold value of the anomaly decision is also relatively reasonable.

V. CONCLUSION AND FUTURE WORKS

In this paper, we defined a time-series point anomaly and proposed an unsupervised time-series anomaly detection model, MCRAAD, for the timely detection of emergencies. We captured the correlation between time series by calculating the feature matrix of subsequences in a sliding window and made our model more robust by taking advantage of an autoencoder. Then, we used a multilayer convolution encoder to capture the characteristics of the feature matrix sequence and obtain the temporal pattern through ConvLSTM networks. Finally, MCRAAD predicted the self-feature matrix at every time step according to the reconstructed feature matrix sequence decoded by the convolutional decoder. The abnormal score of each time step was calculated to determine whether this step is abnormal. We performed several experiments with our model and seven basic models on synthetic datasets and real housing monitoring datasets. The results showed that MCRAAD performs better than these basic models. Even with the increasing noise, the performance of MCRAAD is always superior to the other seven basic models.

Although our approach in both simulated data and real datasets achieved good performance, due to the influence of many unpredictable factors in reality, we cannot obtain completely normal data as the training set. In this work, with the increase of noise, the anomaly detection performance of the proposed framework is always in a good state, but the detection performance drops sharply. In our future work, we will develop a threshold-setting strategy and design an anomaly detection model that does not require a perfectly normal training set. Another goal is to add attention mechanism to the model to build a noise-insensitive framework for multivariate time series anomaly detection.

REFERENCES

[1] M. Canizo, I. Triguero, A. Conde, and E. Onieva, "Multi-head CNN-RNN for multi-time series anomaly detection: An industrial case study," *Neurocomputing*, vol. 363, pp. 246–260, Oct. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231219309877

[2] M. Gupta, J. Gao, Y. Sun, and J. Han, "Community trend outlier detection using soft temporal pattern mining," in *Machine Learning and Knowledge Discovery in Databases* (Lecture Notes in Computer Science), vol. 7524, P. A. Flach, T. D. Bie, and N. Cristianini, Eds. Bristol, U.K.: Springer, Sep. 2012, pp. 692–708.

[3] D. Birant and A. Kut, "Spatio-temporal outlier detection in large databases," *J. Comput. Inf. Technol.*, vol. 14, no. 4, pp. 291–297, 2006. [Online]. Available: http://cit.srce.hr/index.php/CIT/article/view/1613

[4] E. J. Keogh, J. Lin, and A. W. Fu, "HOT SAX: Efficiently finding the most unusual time series subsequence," in *Proc. 5th IEEE Int. Conf. Data Mining (ICDM)*. Houston, TX, USA, Nov. 2005, pp. 226–233, doi: 10.1109/ICDM.2005.79.

[5] H. Wang, M. J. Bah, and M. Hammad, "Progress in outlier detection techniques: A survey," *IEEE Access*, vol. 7, pp. 107964–108000, 2019.

[6] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, p. 15, 2009, doi: 10.1145/1541880.1541882.

[7] I. Yaqoob, E. Ahmed, I. A. T. Hashem, A. I. A. Ahmed, A. Gani, M. Imran, and M. Guizani, "Internet of Things architecture: Recent advances, taxonomy, requirements, and open challenges," *IEEE Wireless Commun.*, vol. 24, no. 3, pp. 10–16, Jun. 2017, doi: 10.1109/MWC.2017.1600421.

[8] N. Jin, Y. Zeng, K. Yan, and Z. Ji, "Multivariate air quality forecasting with nested LSTM neural network," *IEEE Trans. Ind. Informat.*, early access, Mar. 17, 2021, doi: 10.1109/TII.2021.3065425.

[9] H. Liang, L. Song, J. Wang, L. Guo, X. Li, and J. Liang, "Robust unsupervised anomaly detection via multi-time scale DCGANs with forgetting mechanism for industrial multivariate time series," *Neurocomputing*, vol. 423, pp. 444–462, Jan. 2021.

[10] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," 2019, *arXiv:1901.03407*. [Online]. Available: http://arxiv.org/abs/1901.03407

[11] A. Fernández, S. Garcia, M. Galar, C. R. Prati, B. Krawczyk, and F. Herrera, *Learning from Imbalanced Data Sets*. Cham, Switzerland: Springer, 2018, pp. 1–377.

[12] A. Bayati, K. K. Nguyen, and M. Cheriet, "Multiple-Step-Ahead traffic prediction in high-speed networks," *IEEE Commun. Lett.*, vol. 22, no. 12, pp. 2447–2450, Dec. 2018.

[13] P. Xiang, H. Zhou, H. Li, S. Song, W. Tan, J. Song, and L. Gu, "Hyperspectral anomaly detection by local joint subspace process and support vector machine," *Int. J. Remote Sens.*, vol. 41, no. 10, pp. 3798–3819, May 2020, doi: 10.1080/01431161.2019.1708504.

[14] J. Ma and S. Perkins, "Time-series novelty detection using one-class support vector machines," in *Proc. IJCNN*, vol. 3. IEEE, 2003, pp. 1741–1745.

[15] T. F. Liu, M. K. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," *ACM Trans. Knowl. Discovery Data*, vol. 6, no. 1, pp. 1–39, 2012.

[16] Y. Qin, D. Song, H. Cheng, W. Cheng, G. Jiang, and W. G. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," in *Proc. IJCAI*, 2017, pp. 2627–2633.

[17] W. Cheng, K. Zhang, H. Chen, G. Jiang, and W. Wang, "Ranking causal anomalies via temporal and dynamical analysis on vanishing correlations," in *Proc. KDD*, 2016, pp. 805–814.

[18] S. Aminikhanghahi and D. J. Cook, "A survey of methods for time series change point detection," *Knowl. Inf. Syst.*, vol. 51, no. 2, pp. 339–367, May 2017.

[19] M. Hu, X. Feng, Z. Ji, K. Yan, and S. Zhou, "A novel computational approach for discord search with local recurrence rates in multivariate time series," *Inf. Sci.*, vol. 477, pp. 220–233, Mar. 2019, doi: 10.1016/j.ins.2018.10.047.

[20] N. Ho, H. Vo, M. Vu, and T. B. Pedersen, "AMIC: An adaptive information theoretic method to identify multi-scale temporal correlations in big time series data," *IEEE Trans. Big Data*, vol. 7, no. 1, pp. 128–146, Mar. 2021.

[21] T. Chen, X. Liu, B. Xia, W. Wang, and Y. Lai, "Unsupervised anomaly detection of industrial robots using sliding-window convolutional variational autoencoder," *IEEE Access*, vol. 8, pp. 47072–47081, 2020.

[22] D. Xiao-Chao and Y. Lin, "Traffic flow prediction based on multivariate linear AR model," *Comput. Eng.*, vol. 38, no. 1, pp. 90–92, and 95, 2021.

[23] R. Patel and G. Saha, "Time series regression of weather parameters over the last century for cotton crop," *Proc. Int. Conf. Intell. Syst. Signal Process.*, 2018, pp. 125–153.

[24] Y. Wang, K. Huang, and T. Tan, "Group activity recognition based on ARMA shape sequence modeling," in *Proc. ICIP*, p. 209, 2007.

[25] T. Ergen and S. S. Kozat, "Unsupervised anomaly detection with LSTM neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 8, pp. 3127–3141, Aug. 2020.

[26] B. Lindemann, N. Jazdi, and M. Weyrich, "Detektion von anomalien zur Qualitätssicherung basierend AUF sequence-to-sequence LSTM netzen," *at-Automatisierungstechnik*, vol. 67, no. 12, pp. 1058–1068, Nov. 2019, doi: 10.1515/auto-2019-0076.

[27] Z. Li, J. Li, Y. Wang, and K. Wang, "A deep learning approach for anomaly detection based on SAE and LSTM in mechanical equipment," *Int. J. Adv. Manuf. Technol.*, vol. 103, nos. 1–4, pp. 499–510, Jul. 2019.

[28] S. Zhang, Q. Tang, Y. Lin, and Y. Tang, "Fault detection of feed water treatment process using PCA-WD with parameter optimization," *ISA Trans.*, vol. 68, pp. 313–326, May 2017.

[29] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. 25th Int. Conf. Mach. Learn. (ICML)*, 2008, pp. 1096–1103.

[30] A. Borghesi, A. Bartolini, M. Lombardi, M. Milano, and L. Benini, "A semisupervised autoencoder-based approach for anomaly detection in high performance computing systems," *Eng. Appl. Artif. Intell.*, vol. 85, pp. 634–644, Oct. 2019.

[31] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery," in *Information Processing in Medical Imaging* (Lecture Notes in Computer Science), vol. 10265, M. Niethammer, M. Styner, S. R. Aylward, H. Zhu, I. Oguz, P. Yap, and D. Shen, Eds. Boone, NC, USA: Springer, Jun. 2017, pp. 146–157.

[32] S. Plakias and Y. S. Boutalis, "Exploiting the generative adversarial framework for one-class multi-dimensional fault detection," *Neurocomputing*, vol. 332, pp. 396–405, Mar. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092523121831498X

[33] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst., Annu. Conf. Neural Inf. Process. Syst.*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Montreal, QC, Canada, Dec. 2014, pp. 2672–2680.

[34] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, and N. V. Chawla, "A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, Jul. 2019, pp. 1409–1416, doi: 10.1609/aaai.v33i01.33011409.

[35] S. Lin, R. Clark, R. Birke, S. Schonborn, N. Trigoni, and S. Roberts, "Anomaly detection for time series using VAE-LSTM hybrid model," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 4322–4326.

[36] D. Song, N. Xia, W. Cheng, H. Chen, and D. Tao, "Deep R-th root of rank supervised joint binary embedding for multivariate time series retrieval," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 2229–2238, doi: 10.1145/3219819.3220108.

[37] D. Hallac, S. Vare, S. Boyd, and J. Leskovec, "Toeplitz inverse covariance-based clustering of multivariate time series data," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Halifax, NS, Canada, Aug. 2017, pp. 215–223.

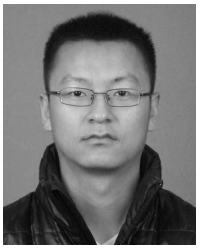
[38] T. Pan, B. Wang, G. Ding, and J. Yong, "Fully convolutional neural networks with full-scale-features for semantic segmentation," in *Proc. 31st AAAI Conf. Artif. Intell.*, S. P. Singh and S. Markovitch, Eds. San Francisco, CA, USA: AAAI Press, Feb. 2017, pp. 4240–4246. [Online]. Available: http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14488

[39] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Proc. Adv. Neural Inf. Process. Syst., Annu. Conf. Neural Inf. Process. Syst.*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Montreal, QC, Canada, Dec. 2015, pp. 802–810.

[40] C. Willmott and K. Matsuura, "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance," *Climate Res.*, vol. 30, no. 1, pp. 79–82, 2005.

[41] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han, "Outlier detection for temporal data: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 9, pp. 2250–2267, Sep. 2014.

[42] V. Pham, N. Nguyen, J. Li, J. Hass, Y. Chen, and T. Dang, "MTSAD: Multivariate time series abnormality detection and visualization," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Los Angeles, CA, USA, Dec. 2019, pp. 3267–3276.
[43] F. Kadri, F. Harrou, S. Chaabane, Y. Sun, and C. Tahon, "Seasonal ARMA-based SPC charts for anomaly detection: Application to emergency department systems," *Neurocomputing*, vol. 173, pp. 2102–2114, Jan. 2016, doi: 10.1016/j.neucom.2015.10.009.
[44] X. Hua, Y. Cheng, H. Wang, Y. Qin, and Y. Li, "Geometric means and medians with applications to target detection," *IET Signal Process.*, vol. 11, no. 6, pp. 711–720, Aug. 2017.
[45] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Proc. Adv. Neural Inf. Process. Syst., Annu. Conf. Neural Inf. Process. Syst.*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., Long Beach, CA, USA, Dec. 2017, pp. 971–980.
[46] P. D. Kingma and L. J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–15.



PEIHAI ZHAO (Member, IEEE) received the B.S. and Ph.D. degrees from Tongji University, Shanghai, China, in 2011 and 2017, respectively.

He is currently a Lecturer with the School of Computer Science and Technology, Donghua University. He has authored about 30 technical papers in journals and conference proceedings, including IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, IEEE/CAA JOURNAL OF AUTOMATICA SINICA, and *Neurocomputing*. He holds over ten patents, including two international patents. His current research interests include data mining, time-series analysis, deep learning, artificial intelligence, and anomaly detection. He is a very active reviewer of many international journals.

XIAOYAN CHANG received the B.E. degree from Henan University of Urban Construction, Henan, China, in 2019. She is currently pursuing the M.S. degree with Donghua University, Shanghai, China. Her current research interests include deep learning, anomaly detection, and time-series representation learning.

MIMI WANG (Member, IEEE) received the M.S. degree in mathematics and applied mathematics from Anhui University of Science and Technology, Huainan, China, and the Ph.D. degree in computer science and technology from Tongji University, Shanghai, China, in 2013 and 2019, respectively. She is currently a Lecturer with the College of Information Science and Technology, Donghua University. She has authored about 40 technical papers in journals and conference proceedings. Her research interests include diagnosability analysis, matrix analysis, formal verification of software, artificial intelligence, and anomaly detection. She is a very active reviewer of many international journals.