Anomaly Detection Based on Convolutional Recurrent Autoencoder for IoT Time Series

Chunyong Yin, Sun Zhang, Jin Wang, *Member, IEEE*, and Neal N. Xiong, *Senior Member, IEEE*

***Abstract*—Internet of Things (IoT) realizes the interconnection of heterogeneous devices by the technology of wireless and mobile communication. The data of target regions are collected by widely distributed sensing devices and transmitted to the processing center for aggregation and analysis as the basis of IoT. The quality of IoT services usually depends on the accuracy and integrity of data. However, due to the adverse environment or device defects, the collected data will be anomalous. Therefore, the effective method of anomaly detection is the crucial issue for guaranteeing service quality. Deep learning is one of the most concerned technology in recent years which realizes automatic feature extraction from raw data. In this article, the integrated model of the convolutional neural network (CNN) and recurrent autoencoder is proposed for anomaly detection. Simple combination of CNN and autoencoder cannot improve classification performance, especially, for time series. Therefore, we utilize the two-stage sliding window in data preprocessing to learn better representations. Based on the characteristics of the Yahoo Webscope S5 dataset, raw time series with anomalous points are extended to fixed-length sequences with normal or anomaly label via the first-stage sliding window. Then, each sequence is transformed into continuous time-dependent subsequences by another smaller sliding window. The preprocessing of the two-stage sliding window can be considered as low-level temporal feature extraction, and we empirically prove that the preprocessing of the two-stage sliding window will be useful for high-level feature extraction in the integrated model. After data preprocessing, spatial and temporal features are extracted in CNN and recurrent autoencoder for the classification in fully connected networks. Empiric results show that the proposed model has better performances on multiple classification metrics and achieves preferable effect on anomaly detection.**

Manuscript received November 16, 2019; accepted January 17, 2020. Date of publication February 7, 2020; date of current version December 17, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61772282, Grant 61772454, Grant 61811530332, and Grant 61811540410, in part by the Priority Academic Program Development of Jiangsu Higher Education Institutions, Postgraduate Research and Practice Innovation Program of Jiangsu Province under Grant KYCX18_1032, in part by the Natural Science Foundation of Jiangsu Province under Grant BK20150460, in part by the Jiangsu Collaborative Innovation Center on Atmospheric Environment and Equipment Technology, and in part by the Key Laboratory of Broadband Wireless Communication and Sensor Network Technology (Nanjing University of Posts and Telecommunications), Ministry of Education. This article was recommended by Associate Editor H. Zhu. *(Corresponding author: Jin Wang.)*

Chunyong Yin and Sun Zhang are with the School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing 210044, China.

Jin Wang is with the School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha 410004, China (e-mail: jinwang@csust.edu.cn).

Neal N. Xiong is with the Department of Mathematics and Computer Science, Northeastern State University, Tahlequah, OK 74464 USA.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSMC.2020.2968516.

Digital Object Identifier 10.1109/TSMC.2020.2968516

***Index Terms*—Anomaly detection, deep learning, Internet of Things (IoT), time series.**

I. Introduction

INTERNET of Things (IoT), as a novel network system integrated with computer, control, and communication technologies, has a significant impact on society and economy [1], [2]. Rely on wireless sensor network (WSN) technology, ubiquitous sensors are distributed in the real environment, collect data from the surrounding area and transmit data to the processing center via mobile communication networks or Internet [3], [4]. Exploiting cloud computing and big data technology, processing center can extract valuable information these data and provide better services. The technology of IoT has been widely applied in medical, intelligent transportation, smart home, and various fields. For example, in the vehicular network, GPS sensors installed on vehicles are exploited to record real-time location information of vehicles [5], and control center can provide users with recommendations, such as shopping malls and scenic spots around them, or to plan more convenient routes for avoiding traffic congestion [6], [7]. In intelligent medical, the status of patients is monitored in real-time through intelligent wearable devices [8]. In the field of security monitoring, via monitoring equipment installed in each corner of the city or the house, it can provide residents with life security, and apply computer vision technology to send out warning signal in abnormal situations [9].

Although IoT plays an important role in various fields, the quality of service depends on the accuracy of sensing data. However, heterogeneous sensors are usually deployed in the harsh and extreme natural environment. For example, a large number of low-cost sensors are directly deployed by the aircraft for monitoring forest or planting areas [10]. In the collection of meteorological information, precipitation and temperature sensors are vulnerable to atrocious weather [11]. Therefore, harsh natural environment and sensor hardware problems could cause the generation of anomaly data which will affect the accuracy and quality of IoT services.

Li and Hong [12] defined anomaly data as follows. Due to the uncertainty of the sensing area and the limited resources of sensors, it is vulnerable to the interference and destruction of external factors, or the impact of external environmental emergencies. The collected data of WSN are likely to have significant difference with actual measurement and the data with deviation are considered as anomalies as shown in Fig. 1.

2168-2216 © 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Southern University of Science and Technology. Downloaded on October 19,2023 at 13:17:59 UTC from IEEE Xplore. Restrictions apply.

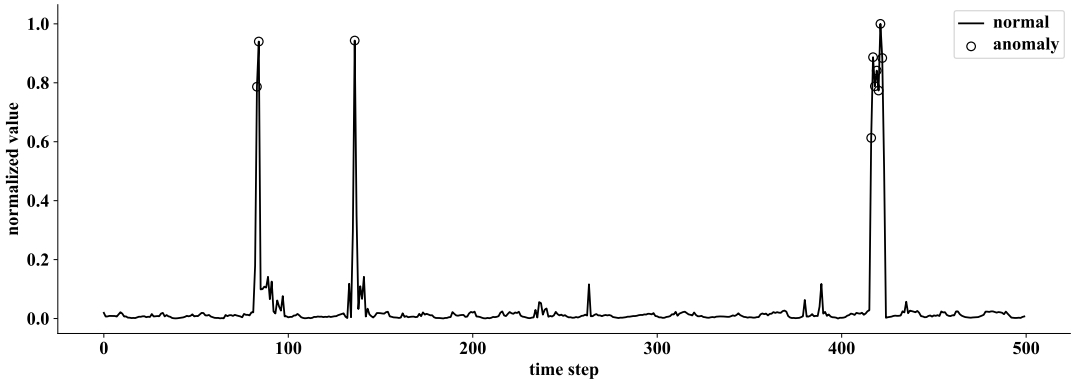

Fig. 1. Example of anomaly points. The circular marker denotes the location of anomaly point of time series which has significant deviation with normal points.

Fei *et al.* [13] summarized the cause of anomaly data into three points.

1) Specific events occurred in the area monitored by sensor nodes (when forest fire occur, the temperature readings of sensors will rise sharply).
2) Sensors may not work normally due to their own hardware failures or energy depletion.
3) The deviation or measurement error exists in collected data due to the influence of external factors (e.g., illumination intensity readings of sensor nodes in shaded areas are significantly lower than those of nodes directly exposed to sunlight).

The anomalous data caused by specific events means that the environment of the monitoring area has changed significantly which is necessary to send out warning signal and take measures to deal with it in time. The anomalous data caused by sensor hardware failure or energy exhaustion reflects the problems of sensor networks which needs to redeploy nodes. The data with measurement errors could affect the service quality and accuracy of IoT which needs to be detected and processed effectively.

The data generated from sensors usually consists of time series which have distinct temporal and sequential characteristics. Therefore, special methods of anomaly detection for IoT time series are necessary. Giannoni *et al.* [14] summarized the conventional methods of anomaly detection.

1) *Statistical-Based Method:* The statistical model is constructed by extracted statistical features from historical measurements. Anomaly values are identified if it mismatches with the statistical model. Low-high pass filter [15] detects all outliers using the historical value of mean and standard deviation. This method is very simple, but it is difficult to detect anomaly data effectively because of low accuracy and recall.
2) *Probability-Based Method:* This method assumes normal observations meet a specific probability density distribution and uses likelihood value as the metric for anomaly detection. Hidden Markov models [16] and Bayes nets [17], [18] are typical probabilistic methods, but the estimation of parameters is complex and time-consuming.
3) *Similarity-Based Method:* It relies on calculating the distance or similarity between normal and anomaly data. Local outlier factor [19] and K-means [20] both use the distance between labeled and unlabeled data for anomaly detection. This method is simple but not suitable for high-dimension data, because the calculation of distance is usually proportional to the number of features.
4) *Prediction-Based Method:* This method exploits historical measurements to build the prediction model that could predict the data at the next time step. If the difference between the real measurement and the predicted value is significant, it will be considered as anomalous. One-class SVM [21] and other regression-based methods rely on accurate prediction to detect anomaly data. However, the prediction-based method can only be applied to time series with periodicity or seasonality. Malhotra *et al.* [22] and Park and Yun [23] have proved that prediction-based method is unsuitable for unpredictable time series.

Although the above anomaly detection methods are based on conventional machine learning algorithms, recently researchers try to apply deep learning to anomaly detection, with the excellent performance of deep learning methods in image, speech signal, and natural language processing. The most representative methods of deep learning are convolutional neural network (CNN) [24] and recurrent neural network (RNN) [25]. The former is mainly applied in the field of image processing, using multiple convolution kernels to extract local features and spatial information from images. The latter is applied in the speech signal and natural language processing, employing memory units to extract temporal information from sequential data. There are diverse implementations of RNN memory units, the most famous of that are long-short term memory (LSTM) [26] and gate recurrent unit (GRU) [27]. LSTM and GRU employ different gate units to overcome the problem of gradient vanishment caused by the memory loss for long term sequences.

Since CNN and RNN can, respectively, extract spatial and temporal features from data, researchers [28] attempt to combine these two methods for better classification and prediction in time series. Kim and Cho [29] first applied C-LSTM [30] model for time series anomaly detection. C-LSTM model consists of CNN, LSTM, and deep neural network (DNN). First, a sliding window is applied to divide the univariate time series into several fixed-length sequences and build the

Authorized licensed use limited to: Southern University of Science and Technology. Downloaded on October 19,2023 at 13:17:59 UTC from IEEE Xplore. Restrictions apply.

structural dataset. Then, high-level spatial and temporal features are extracted from the window data by CNN and LSTM. Finally, the extracted features are input into the fully connected DNN to realize the classification. Based on the research work in [29], we propose the improved network architecture for better anomaly detection. After building dataset, another smaller sliding window is employed to generate time-dependent subsequences. In this way, the features extracted by CNN could have potential temporal relevance which make it more effective for temporal feature extraction in LSTM. Notice that one-dimension convolution layer is employed in our model for better comparing with other methods. One-dimension convolution layer is widely applied in sequence processing and could be considered as two-dimension convolution layer with the height of 1. The features extracted by CNN will be input into the autoencoder with two LSTM layers, and the learning target of autoencoder is to reconstruct features that are more conducive for the classification in DNN. Experiments on Yahoo Webscope S5 Dataset (https://webscope.sandbox.yahoo.com) prove that our neural network architecture can achieve higher scores on various metrics and better performance for anomaly detection. The main contributions of this article are as following.

1) The integrated model of CNN and LSTM-based autoencoder is proposed for time series anomaly detection. With the same idea as C-LSTM, the proposed model combines the advantages of CNN and LSTM. One-dimension convolution layer is preserved, but the single LSTM layer is replaced by LSTM-based autoencoder. The outputs of encoder are constructed by hidden states in each time step which are input into decoder for features extraction. From the evaluation of several classification metrics, the proposed model could have better performances than C-LSTM.
2) Sliding windows are exploited for time series preprocessing. Empiric results show that the sliding window cannot improve the performance of CNN, but it could be effective for the integrated model of CNN and LSTM. This phenomenon implicity prove that the sliding window may bring time dependencies into the features extracted by CNN which could be helpful for temporal feature extraction in LSTM.
3) The possibility of anomaly detection based on reconstruction error is evaluated in our experiments. An autoencoder is only trained by normal data and the target is to learn the distribution of normal data. Considering the distribution difference between normal and anomaly data, the reconstruction error of anomaly data could be higher than that of normal data. However, the results on the Yahoo Webscope S5 dataset show it is difficult to detect anomalies based on reconstruction errors. We think this results might be caused by preprocessing and will be investigated in our future work.

The remainder of this article is organized as follows. In Section II, we review some related work about anomaly detection of time series and deep learning. We introduce our proposed neural network architecture in Section III. Section IV shows related experimental results and compares the impact of different architecture. In Section V, we conclude this article and introduce our research plan in future work.

II. Related Work

A survey of time series data mining is proposed in 2012 [31]. It provides a comprehensive introduction and deep understanding of data mining tasks for time series. In this survey, the time series is defined as a collection of values which are obtained from continuous measurements over time. The major tasks relevant to time series are categorized into seven tasks, including query by content, clustering, classification, prediction, segmentation, motif discovery, and anomaly detection.

1) *Query by Content:* This is a popular research area for time series mining. The task of querying by content aims to retrieve a set of solutions which are the most similar results to a query from users. It relies on the matching between time series and the matching methods can be categorized as whole series matching and subsequence matching.
2) *Clustering:* The task of clustering aims to discover potential natural groups and clusters. The grouping process is to maximize the variance between clusters and minimize the variance within each cluster.
3) *Classification:* Given a set of time series with labels, the task of classification is to train a classifier that can assign correct labels to unseen time series. Training samples are utilized to discover distinctive features distinguishing classes from each other.
4) *Prediction:* It is a concerned issue for diverse research fields, such as economic, meteorology, and security. For long and smooth time series, it is feasible to predict future measurements according to a range of historical observations. The task of prediction is to model temporal dependencies and forecast the next single step or multisteps values.
5) *Segmentation:* The task of segmentation is to generate an approximation of origin time series by reducing the length of time series while remaining vital information. It can be considered as the process of feature reduction and the target is to minimize the reconstruction error between reduced and origin time series.
6) *Motif Discovery:* It aims to find the sequences that appears recurrently in origin time series and it is similar to frequent itemset mining. The idea of motif discovery comes from the gene analysis in bioinformatics and sequences appearing recurrently may have valuable meanings and rules.
7) *Anomaly Detection:* The anomaly detection for time series is defined as finding abnormal sequences in a series [31] and the range of abnormal sequences is relevant to the length of time window. The common method is modeling normal behaviors and find out all sequences deviating from the model.

Although the objectives of time series mining tasks are different, there are three concerns comment to most mining tasks, including representation methods, similarity measurements,

Authorized licensed use limited to: Southern University of Science and Technology. Downloaded on October 19,2023 at 13:17:59 UTC from IEEE Xplore. Restrictions apply.

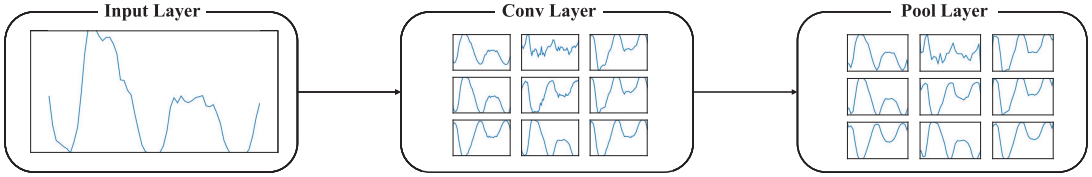

Fig. 2. Extracted features of convolutional and pool layer. The convolutional layer extracts spatial features from input sequence and pool layer realizes feature reduction by down-sampling.

and indexing techniques. Considering the high dimension of time series, it is crucial to reduce the dimensions while remaining necessary information by efficient representation methods. Given the representation method, the similarity or distance between each sequence in time series should be carefully defined for better classification and clustering. Indexing techniques are utilized to manage and retrieve in massive databases.

As the introduction in Section I, Giannoni *et al.* [14] summarized the application of conventional machine learning algorithms in anomaly detection. Similarly, Kim and Cho [29] proposed that anomaly detection is the process of extracting features from data and identifying their normal and abnormal patterns. From the perspective of feature extraction, they divide the recent research on anomaly detection into three categories: 1) statistical modeling; 2) temporal feature extraction; and 3) spatial feature extraction. Whether using the prediction error of time series as the detection target [32], or reconstructing features to increase the difference between normal and anomaly data [22], these anomaly detection methods can be regarded as extracting the various necessary information from the original data for better classification. Most existing studies only consider to extract spatial or temporal features separately. Therefore, in the paper [33]–[35], CNN and LSTM are combined to extract high-level information. In the remaining part of this section, we review the background and application of CNN and RNN.

CNN is proposed by Hubel and Wiesel [36] in 1968 according to their study of the visual cortex for cats and monkeys. They find that there are neurons in the visual cortex that response individually to small regions of the vision field. The definition of the receptive field is the particular region of sensory space in which a stimulus will modify the firing of that neuron. Adjacent neurons have similar and overlapping receptive field. The size and location of the receptive field changes systematically between visual cortex, resulting in a complete visual-spatial map. The studies on the receptive field provide theoretical basis for the local perception of CNN. CNN consists of input layer, hidden layer, and output layer. The hidden layer contains convolution layer, pool layer, activation layer, and full connection layer. As the core of CNN, convolution layer, which is inspired by receptive field, calculates the convolution of the data from input layer with filters or kernels to extract high-level spatial features. The main function of pool layer is down-sampling which means to reduce the number of features. Because the features extracted by adjacent filters in the convolution layer are similar. In order to reduce the size of data while retaining important information, pool layer selectively discards unnecessary features.

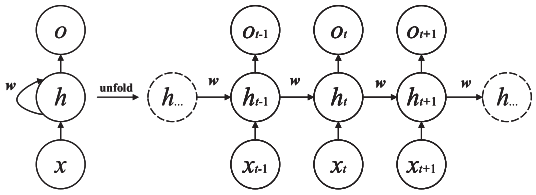

Fig. 3. Architecture of RNN. The neurons in hidden layer are not only connected with input and output layer but also neurons in the same layer for preserving historical memory.

Fig. 2 shows the spatial features extracted from input data in the convolution layer and the process of down-sampling in the pool layer. The input data is a time series containing 60 values. The kernel size of nine filters employed in the convolution layer is three and zero padding are applied to keep the length of the extracted features unchanged. In the convolution layer of Fig. 2, the features extracted by adjacent filters have similar and overlapping regions, therefore, the max-pool layer is applied for down-sampling the extracted features. The kernel size in the max-pool layer is 2, and the stride length is also set to 2, so the sequence length after down-sampling is 30. In the pool layer of Fig. 2, spatial information after down-sampling is preserved while the number of features is reduced.

The excellent effect of spatial features extraction in CNN make researchers apply it to time series. However, CNN is usually employed for processing images and the convolution kernel is two-dimension. Time series is a sequence of univariate or multivariates and it needs to be transformed into special formats. Wang *et al.* [37] proposed to map univariable network traffic into a two-dimension image and features of the image are extracted by CNN. Liu *et al.* [38] proposed a tensor scheme along with a multivariate CNN for classifying multivariate time series. These methods efficiently extract spatial features from time series and improve the performances of different classifiers. But Kim and Cho [29] pointed out that temporal information loss may occur in the convolution and pooling layers when extracting spatial features from time series.

RNN is a type of neural network suitable for processing sequential data. Generally, neural network is connected by weights between layers. The most distinct characteristic of RNN is that neurons in the same layer are also connected by weights. As shown in Fig. 3, RNN consists of input layer, hidden layer, and output layer. The state of the hidden layer could change over time. In the hidden layer, neurons are not only connected with the input layer and output layer but also neurons located in the same hidden layer. The hidden state at the current time t is generated by new input data at time t and

Authorized licensed use limited to: Southern University of Science and Technology. Downloaded on October 19,2023 at 13:17:59 UTC from IEEE Xplore. Restrictions apply.

the hidden state at the previous time $t-1$. RNN preserves part of the input data at each time step as the memory of historical information by employing hidden units. However, because of the limited capacity of hidden units, it may cause the loss of historical information which could raise the problem of gradient vanishment when a long sequence is input.

The proposed architectures of LSTM and GRU solve the problem of long-term memory in RNN by employing different gate units to control the memory of new information and the forgetting of historical information. Cell state is applied in LSTM to preserve long-term memory, in which historical memory is controlled by input, forget, and output gate. Input and forget gate are designed to control the updating and deleting of long-term memory maintained in cell state, while the output gate is exploited to control the output information from the cell state. GRU, as a variant of LSTM, has simpler architecture by only consists of update and reset gate. Reset gate determines whether to forget preserved hidden state information, and it can be regarded as the combination of forget and input gate in LSTM. Update gate decides whether to update the hidden state to the new one, which is similar to the output gate of LSTM.

The characteristic of temporal features extraction in RNN makes it popular in time series mining. Karim *et al.* [39] proposed the integrated model of LSTM and CNN with attention mechanism for time series classification. From the experimental results, LSTM improve the performance of the fully convolutional network and attention mechanism detects the regions of the input sequence that related to the class label. Malhotra *et al.* [32] employed LSTM for time series prediction and realize anomaly detection by prediction errors. LSTM extracts temporal information from historical measurements and stores in hidden units. The features of hidden units are input into the fully connected network to make multistep prediction. Then, it builds the Gaussian distribution model from prediction errors and utilizes likelihood probability as anomaly detection score.

Autoencoder belongs to unsupervised the neural network model [40]. It could extract hidden features from input data and reconstruct original data with these features. Autoencoder can be applied not only for dimension reduction but also for further classification and prediction. As shown in Fig. 4, the autoencoder usually consists of an encoder and a decoder. The encoder encodes the input raw data into the intermediate vector C which is parsed into the original data or target data in the decoder. Fig. 4(a) shows the autoencoder implemented by fully connected neural networks. The information preserved in hidden neurons is considered as the encoded features. In the field of machine translation, autoencoder is trained to extract semantics from input sentences, and then reconstruct the target language from the extracted semantics. The input sentences are in the format of sequential data, and temporal correlation exists in words. To realize encoding and decoding for sequential data, Sutskever *et al.* [41] proposed the RNN-based autoencoder model as shown in Fig. 4(b). RNN-based encoder receives the input sequence at each time step, extracts and memorizes the temporal features of input data, which is preserved in hidden units as encoded information. RNN-based decoder receives the encoded information and reconstructs the original or target sequence.

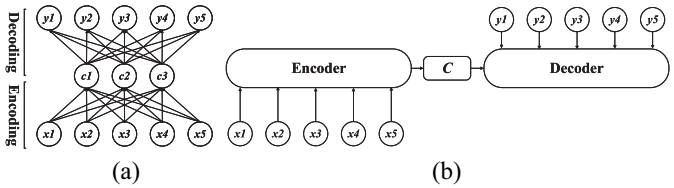

Fig. 4. Architecture of autoencoder. The left one is conventional autoencoder implemented by fully connected networks and the right one is RNN-based autoencoder which is efficient for processing sequential data.

In this article, we combine the capacity of extracting features automatically in autoencoder with the advantage of processing sequential data in LSTM to extract better features for anomaly detection from input time series. Malhotra *et al.* [22] first proposed to apply the LSTM-based autoencoder for anomaly detection in time series, and proves that autoencoder could have better performance for unpredictable data than prediction-based method, but they only use normal data to train the autoencoder, and exploit the error between decoded and original data for anomaly detection. We find that this method is unsuitable for the Yahoo Webscope S5 dataset, because there are only subtle differences between normal and anomaly data. Even if only normal data is used to train autoencoder, the error after decoding is not enough to distinguish between normal and anomaly data. We will design the experiments to prove it in Section IV.

III. Proposed Anomaly Detection Scheme

In this section, we first introduce the notations used in this article and give the statement of the problem we aim to study. Then, the functions and implements of the improved model are detailed. The architecture and hyper parameters are presents at last.

A. Notation and Problem Statement

Consider a time series $X = (x^1, x^2, \ldots, x^n)^T = (x_1, x_2, \ldots, x_t) \in \mathbb{R}^{n\times t}$, where t is the time stamp of each value and n is the number of features. We employ $x_t = (x_t^1, x_t^2, \ldots, x_t^n) \in \mathbb{R}^n$ to denote an input vector at time t. For some research fields, the input series is an univariable series and it has only one feature at each time step. Anomaly detection for time series consists of two problems: 1) finding all anomaly points and 2) labeling the target series. The work in [15], [21], and [42] aim to find all outliers and [19], [20] are trying to label target series correctly. The former is based on regression method and the later belongs to clustering or classification problem. The problem of detecting outliers can be transformed into classification or clustering problem with the utilizing of sliding window [43]. Sliding window could fragment the whole univariable series into continuous shorter sequence and obtain the two-dimension dataset $D = (d_1, d_2, \ldots, d_{t-T+1}) = ((x_1, \ldots, x_T); (x_2, \ldots, x_{T+1}); \cdots; (x_{t-T+1}, \ldots, x_t)) \in \mathbb{R}^{T\times(t-T+1)}$, in which T is the length of sliding window and d_i is consider as anomaly series if it contains any anomaly value from origin series.

Authorized licensed use limited to: Southern University of Science and Technology. Downloaded on October 19,2023 at 13:17:59 UTC from IEEE Xplore. Restrictions apply.

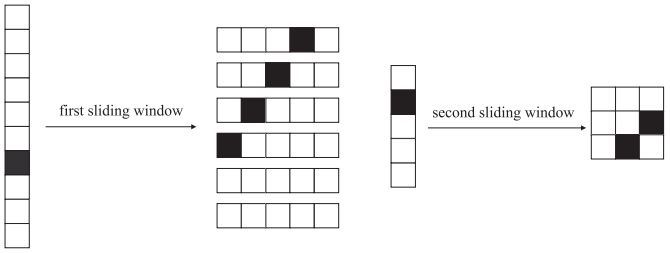

Fig. 5. Diagram of two-stage sliding window. Origin univariable time series is split into several sequences by first sliding window. Each sequence is transformed into a tensor with one channel by second sliding window.

Therefore, the problem of anomaly detection for time series X is transformed to label each input vector for D. We decide to employ classification method to solve the problem and training dataset with labels is utilized for training neural networks. The target can be represented as

$$\max(\text{Prob}(y_i = 0|d_i), \text{Prob}(y_i = 1|d_i)) \tag{1}$$

where $y_i = 1$ denotes d_i is anomalous series and $y_i = 0$ represents d_i is normal series.

B. Model and Scheme

The proposed anomaly detection model is based on the studies in [29] and [30], which consists of CNN, LSTM-based autoencoder and DNN. CNN and LSTM-based autoencoder are employed to extract high-level spatial and temporal features from time series. DNN composed of forward fully connected neural networks is applied to classify extracted features and realize anomaly detection. The details of CNN and LSTM-based autoencoder are as follows.

Origin time series with anomalous points in the Yahoo Webscope S5 dataset are split into several sequences by first sliding window. If a sequence contains anomalous point, it would be considered as anomalous. Then, each sequence is transformed and unsqueezed into continuous fragments as the input of CNN by the second sliding window, as shown in Fig. 5. Our experiments only focus on univariable time series and each sequence would be transformed into three-dimension tensor with one channel. Considering heterogeneous resources are connected in IoT which means IoT time series are multivariable, our method is also compatible and multivariable time series will be preprocessed as three-dimension tensor with multichannels.

The architecture of CNN in our improved model is similar to that in C-LSTM. It consists of one-dimension convolution layer, one-dimension max-pool layer, and the relu function exploited as the activation function. Convolution layer could extract high-level spatial features, but extracted information by adjacent convolution kernels are similar and redundant. Therefore, the max-pool layer is applied for down-sampling while maintaining spatial information. The example of the convolution and pool layer is shown in Fig. 2. Time series need to be preprocessed by sliding window before being input into convolution layer and one-dimension univariable time series is transformed into the two-dimension dataset. Different from C-LSTM, we employ two-stage sliding window to preprocess time series. The first sliding window is same as that in C-LSTM, but the second smaller sliding window transforms each sequence in the two-dimension dataset into continuous time-dependent fragments as basic temporal feature extraction. In the experiments of Section IV, we evaluate the impacts of single and two-stage sliding window. The results prove that two-stage sliding window are more efficient for the combined model of CNN and LSTM.

The most distinct of improved model is that LSTM is replaced by LSTM-based autoencoder, which is inspired by the studies in [22] and [23]. For RNN-based autoencoder, time series $x = (x_1, x_2, \ldots, x_t) \in \mathbb{R}^{n\times t}$ is input into the encoder, and the hidden state of encoder will be updated by

$$h_t = f_1(h_{t-1}, x_t) \tag{2}$$

where $h_t \in \mathbb{R}^m$ is the hidden state of encoder at time step t. m is the size of hidden units and f_1 is nonlinear activation function. The basic idea of diverse RNN variants is same, and the difference is various updating methods of hidden state. LSTM employs cell state and hidden state to capture long short-term dependencies under the control of gate units [26]. The cell state s_t is controlled by forget gate f_t, input gate i_t, and output gate o_t [44]. The cell state and hidden state will be updated as follows:

$$f_t = \sigma\left(W_f\left[h_{t-1}; x_t\right] + b_f\right) \tag{3}$$
$$i_t = \sigma\left(W_i\left[h_{t-1}; x_t\right] + b_i\right) \tag{4}$$
$$o_t = \sigma\left(W_o\left[h_{t-1}; x_t\right] + b_o\right) \tag{5}$$
$$s_t = f_t \odot s_{t-1} + i_t \odot \tanh\left(W_s\left[h_{t-1}; x_t\right] + b_s\right) \tag{6}$$
$$h_t = o_t \odot \tanh\left(s_t\right) \tag{7}$$

where $[h_{t-1}; x_t] \in \mathbb{R}^{m+n}$ is the concatenation of previous hidden state h_{t-1} and current input x_t. σ is the activation function and $\odot$ represents element-wise multiplication. The weight W and bias b of forget, input and output gate are parameters to learn. Forget gate decides the remained information of previous cell state and input gate controls the memory of current input data. Then, current hidden state h_i will be computed by current cell state and output information. The hidden state can be considered as encoded information or intermediate information which will be exploited in the decoding process. For prediction purpose, the decoder aims to learn the mapping from h_t to $\tilde{x}_t$, and tries to achieve the minimum difference between $\tilde{x}_t$ and x_t. For classification purpose, the decoder will extract features from hidden states to realize better classification result. Fully connected layer could accomplish the different purpose, but we employ LSTM-based decoder to extract high-level features. The hidden states of each time step will be input into the decoder as new time series. The size of hidden units in decoder is twice more than that of encoder. Then, the hidden state h'_t of decoder will be input into fully connected layer as reconstructed features for better classification performances.

The trained model could be employed to detect anomalous time series as shown in Algorithm 1. First, the raw time series X is transformed into the two-dimension dataset D by the first-stage sliding window and each fragment $d_i \in D$ in the dataset will be detected by the trained model. Continuous

Authorized licensed use limited to: Southern University of Science and Technology. Downloaded on October 19,2023 at 13:17:59 UTC from IEEE Xplore. Restrictions apply.

TABLE I
HYPE PARAMETERS AND ARCHITECTURE OF PROPOSED MODEL

CNN				LSTM-based autoencoder & DNN			
Type	#Filter	Kernel size	Stride	Type	Input size	Hidden size	Output size
Conv1d	64	5	1	LSTM (Encoder)	64	64	-
ReLU	-	-	-	LSTM (Decoder)	64	128	-
MaxPool1d	-	2	2	Linear (DNN)	128	-	510
Conv1d	32	5	1	ReLU (DNN)	-	-	-
ReLU	-	-	-	Linear (DNN)	510	-	2
MaxPool1d	-	2	2	ReLU (DNN)	-	-	-

Algorithm 1 Proposed Anomaly Detection Scheme

Input:
The raw time series $X = (x_1, x_2, x_3, \cdots, x_t)$;
The length of first-stage sliding window, T;
The length of smaller second-stage sliding window, T' ;
The trained model consists of CNN, LSTM-based autoencoder and DNN;

Output:
The label of each fragment in D;

1: Generating two-dimension dataset by sliding window: $D = (d_1, d_2, \ldots, d_{t-T+1})$;
2: **for** each $d_i \in D$ **do**
3: Generating sequences S_i from d_i and the length of each sequence is T';
4: Extracting high-level spatial features from S_i by CNN;
5: Extracting high-level temporal features by LSTM-based autoencoder;
6: Inputting extracted features into DNN for classification;
7: d_i is labeled as anomalous if $Prob(y_i = 1|d_i) > Prob(y_i = 0|d_i)$;
8: **end for**

sequences S will be generated from the fragment d_i by the shorter second-stage sliding window and the sequences could be considered as low-level temporal features which will be fed into the trained model. Then, CNN and LSTM-based autoencoder will extract high-level spatial and temporal features from sequences and extracted features are input into DNN for classification. Finally, DNN will output the probabilities of normal Prob$(y_i = 0|d_i)$ and anomaly Prob$(y_i = 1|d_i)$. If Prob$(y_i = 1|d_i) >$ Prob$(y_i = 0|d_i)$, d_i will be labeled as anomalous, otherwise, it will be labeled as normal.

C. Architecture and Hyper Parameters

Before the training of CNN and LSTM, some hyper parameters need to be set, such as the number of filters in CNN, the size of convolution kernel, stride length, and the number of hidden units in LSTM. These hyper parameters could affect the performance and learning speed of the neural network model. To realize better anomaly detection for time series, we have made improvements and adjustments based on C-LSTM. The architecture and hyper parameters of improved model are shown in Table I.

First, the sliding window of length 60 is applied to transform the one-dimension time series into the two-dimension dataset. Before each sequence in dataset is input into CNN, it will be transformed and unsqueezed into continuous fragments of size $51 \times 1 \times 10$ by smaller sliding window of length 10, in which the length of time step is 51, the number of input channel is 1, and the number of input features is 10. After passing through the first convolution layer, the size of data is $51 \times 64 \times 10$. The convolution layer exploits zero padding to maintain the number of features, but the max-pool layer dose not use any padding. The size of down-sampled data from the max-pool layer is $51 \times 64 \times 5$. Then, the data is input into the second convolution layer and the size of output is $51 \times 32 \times 5$. After the down-sampling in second max-pool layer, the size of output is $51 \times 32 \times 2$.

The output data is flattened as 51×64 before be input into the LSTM-based autoencoder. The number of hidden units in the LSTM-based encoder is set to 64, while that in the decoder is set to 128. After receiving the data at each time step, the size of the hidden state in the encoder is 51×64, which will be input into the decoder as encoded information. The size of the hidden state in the decoder is 51×128 and only the hidden state at last time step is input into the fully connected network for anomaly detection, which means the size of data input into DNN is 1×128. Finally, the features extracted by CNN and LSTM-based autoencoder are input into DNN consisted by two fully connected layers, and the output of fully connected network is two-dimension which is the identification for anomaly detection.

IV. PERFORMANCE ANALYSIS

We use Python3.7 as the programming language and the CUDA version of PyTorch0.4.1 as neural network framework. Because of the limitation of cost and funds, all experiments are executed in the Google Colab cloud platform (https://colab.research.google.com). Google Colab provides a free online programming environment and Tesla K80 GPU for deep learning researchers. To solve the limitation of CUDA memory, mini-batch training is applied in our experiments and the batch size is set to 512. The details of the time series preprocessing and experiments are as follows.

A. Data Preprocessing

Yahoo Webscope S5 dataset is applied in our experiments, which is a labeled anomaly detection dataset. It consists of real and synthetic time series with labeled anomaly points. The whole dataset contains four benchmark subsets and we select the A1 benchmark to evaluate the proposed anomaly detection

Authorized licensed use limited to: Southern University of Science and Technology. Downloaded on October 19,2023 at 13:17:59 UTC from IEEE Xplore. Restrictions apply.

TABLE II
DETAILS OF EACH DATASET

Type	Total	Normal	Anomaly	Ratio
Training	54547	49414	5133	9.41%
Testing	36366	32943	3423	9.41%
Origin	90913	82357	8556	9.41%

model. A1 benchmark has 67 files with 94 866 values, but only 1669 of them are labeled as anomalous. Following the instruction of paper [29], a sliding window is applied to generate continuous fragments and the fragment is considers as anomalous if it contains any anomaly point. In this way, 82 357 normal fragments and 8556 anomaly fragments are created. These fragments constitute the entire dataset and it is divided into training and testing dataset with the ratio of 0.4. Table II shows the details of each dataset and the ratio of anomaly data is unchanged.

In our experiments, anomaly detection is treated as classification problem, and we evaluate different models by accuracy, precision, recall, and F1-score. For binary classification problem, the error is denoted as false positive (FP) if normal instance is labeled as anomalous. False negative (FN) means anomaly instance is labeled as normal. Similarly, true positive (TP) and true negative (TN) represent anomaly and normal instance are identified correctly. Different metrics can be evaluated as follows:

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{8}$$

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{9}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{10}$$

$$f1 - \text{score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \tag{11}$$

B. Results and Analysis

In our anomaly detection model, we exploit the two-stage sliding window to preprocess origin univariate time series. The first sliding window aims to transform one-dimension time series into the two-dimension dataset, while the second sliding window can be considered to extract low-level temporal features that will bring time dependencies to the spatial features extracted by convolutional layer. We design the experiments to evaluate the effect of the two-stage sliding window for the integration of CNN and LSTM.

Four models are evaluated in this experiment, including CNN without sliding window (cnn), CNN with sliding window (cnn + win), and the integration of CNN and LSTM without sliding window (cnn + lstm) and with sliding window (cnn + lstm + win). All the models are trained by training dataset for 500 epochs with the learning rate 1e-2, and the performances on the testing dataset are shown in Fig. 6. The left one shows the loss values of four models in the training process and the right one is testing scores for different metrics. The integration model of CNN and LSTM reaches the lowest training loss value, besides it gets the highest scores

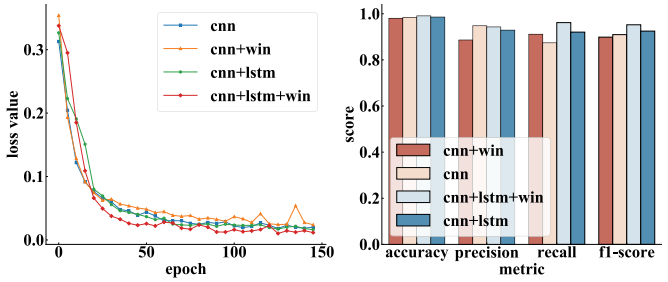

Fig. 6. Effect of two-stage sliding window. The combined model of CNN and LSTM has the lowest loss value and highest scores on four classification metrics after two-stage sliding window preprocessing.

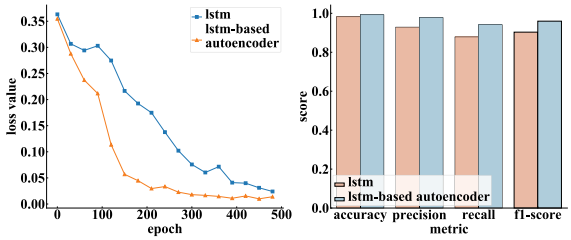

Fig. 7. Comparison between LSTM and LSTM-based autoencoder. LSTM-based autoencoder could reach lower loss value and better performance for classification than LSTM.

in accuracy, recall, and f1-score. It can be found from testing results that two-stage sliding window is useless for single CNN model, but it works well in the combined model of CNN and LSTM. We think that the extracted features of CNN with single sliding window have no temporal relevance which make it difficult to extract temporal features for LSTM. But two-stage sliding window translates each record into multiple continuous fragments and provides time dependencies for better temporal features extraction.

In paper [29], the C-LSTM model consists of CNN and LSTM, where LSTM is applied to extract high-level temporal features. The improved model replaces the single LSTM layer by LSTM-based autoencoder and achieve better temporal feature extraction. We evaluate the classification results of the single LSTM model and LSTM-based autoencoder. Two models are trained by training dataset containing normal and anomaly data for 500 epochs with learning rate 1e-3. As shown in Fig. 7, two models are trained by training dataset for 500 epochs with the learning rate 1e-3. LSTM-based autoencoder can reach the lowest loss value and get best performances on all metrics. This experiment proves that the LSTM-based autoencoder could extract better features for further classification.

Malhotra *et al.* [22] trained LSTM-based autoencoder with only normal data and realizes anomaly detection purpose using the errors between origin data and decoded data. They assume the error vectors are normal distribution and using maximum likelihood estimation to estimate the parameters. However, we find this method is unsuitable for Yahoo Webscope dataset, because there is only small difference between normal and anomaly data. Although the model is trained with only normal data, the reconstruction error is not enough to identify anomaly data correctly. To prove our opinion, the LSTM-based autoencoder is trained by 49 414 normal data in training dataset

Authorized licensed use limited to: Southern University of Science and Technology. Downloaded on October 19,2023 at 13:17:59 UTC from IEEE Xplore. Restrictions apply.

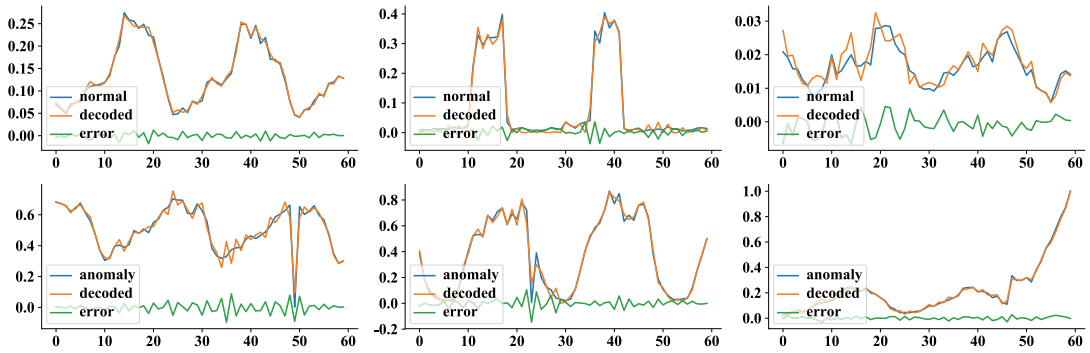

Fig. 8. Reconstruction error of LSTM-based autoencoder. Although the model is trained only by normal data, the decoded data is still pretty similar to origin anomaly data and the reconstruction errors are not distinct for classification.

TABLE III
CLASSIFICATION RESULTS OF RECONSTRUCTION ERROR

Model	Accuracy(%)	Precision(%)	Recall(%)	F1(%)
SVM	90.80	92.47	25.12	4.89
KNN	94.18	85.12	46.30	59.98
DNN	92.97	82.06	32.34	46.40

TABLE IV
COMPARISON OF VARIOUS ANOMALY DETECTION MODELS

Model	Accuracy(%)	Precision(%)	Recall(%)	F1(%)
CNN	98.36	94.80	87.41	90.96
LSTM	98.23	92.93	87.88	90.33
LSTM-AE	99.25	97.84	94.16	95.97
C-LSTM	98.59	92.92	92.05	92.49
C-LSTM-AE	99.62	98.78	97.20	97.98

and reconstruction errors are obtained from testing dataset containing both two categories data.

As shown in Fig. 8, the first row represents the reconstruction error of normal data and the second row is that of anomaly data. The LSTM-based autoencoder is trained only by normal data but the decoded data of anomaly data is still pretty similar to origin data which results in small errors. To evaluate the classification effect of reconstruction error, we employ support vector machine (SVM), k-nearest neighbor (KNN), and DNN to classify on error vectors. SciKit-Learn (https://scikit-learn.org) provides simple and efficient implementation of machine learning algorithms. We directly exploit the models and default parameters from the SciKit-learn module. The classification results of error vectors are shown in Table III. All three models have low F1-scores and high accuracy is caused by unbalance categories. Low F1-score and recall prove it difficult to find out anomaly data from reconstruction errors. Therefore, we decide to train LSTM-based autoencoder with both two categories data and consider it as high-level feature extraction for better classification, rather than expanding the difference between two categories data.

From the above evaluating results, we propose the improved anomaly detection model for time series which is the integration of CNN, LSTM-based autoencoder and DNN. Before time series is input into the improved model, it is preprocessed by two-stage sliding window. Various anomaly detection models for anomaly detection are compared, containing basic CNN, LSTM, LSTM-based autoencoder (LSTM-AE), C-LSTM and our improved model C-LSTM-AE. The experimental results are shown in Table IV, and C-LSTM-AE has the best performances on all metrics.

The processing of sliding window can be considered as low-level temporal information extraction and the length of sliding window T is vital for following convolution operation and classification. We compare the anomaly detection performances under different sliding window lengths and our proposed model achieves best results when sliding window length is 10. As shown in Fig. 9, the performances are worse if the window length is too larger or too small. $T = 60$ means it has not been processed by the second sliding window which could lost details, and the anomalies could be relatively normal in a tiny sliding window like $T = 5$. Therefore, choosing an appropriate window length is meaningful for time series mining, but it may be an empiric task.

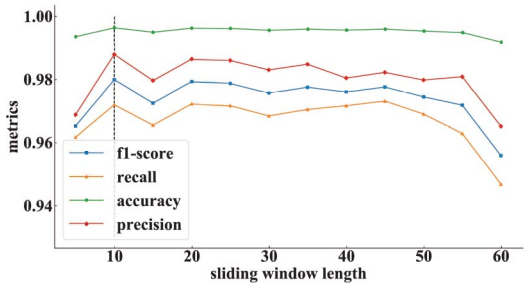

Fig. 9. Comparison results of our proposed model under different sliding window lengths. To choosing optimal sliding window length, we utilize four metrics to evaluate the anomaly detection results under different window lengths and it achieved the best results when the length $T = 10$.

V. CONCLUSION

To achieve better anomaly detection for time series, we propose new architecture combining CNN and LSTM-based autoencoder. Besides, we prove that two-stage sliding window could generate time relevance for better temporal feature extraction in LSTM which receives the output of CNN in the combined model. Two-stage sliding window can be considered as basic temporal feature extraction, CNN could extract high-level spatial features and LSTM is employed to extract high-level temporal features. And we evaluate different

Authorized licensed use limited to: Southern University of Science and Technology. Downloaded on October 19,2023 at 13:17:59 UTC from IEEE Xplore. Restrictions apply.

methods of applying LSTM-based autoencoder in anomaly detection and exploit LSTM-based autoencoder to extract better features for classification in fully connected network. From the experimental results, the proposed model achieves the best accuracy, precision, recall, and F1-score. However, because of the limitation of hardware resources, we think the parameters and architecture of proposed model can be optimized through more trials.

In future work, we intend to introduce attention mechanism and solve the problem of anomaly detection in multivariable time series. Generative adversarial network is also a challenge issue and it has the possibility be applied for anomaly detection. Anomaly detection is only the basis of guaranteeing IoT services and there is much work deserve to study. For example, anomaly events can be divided into three categories: 1) single anomaly event; 2) group anomaly event; and 3) collective anomaly event. It is worth to figure out the type of detected anomaly event. The source of anomaly event is another vital issue and if it is caused by measurement errors, the recovery of anomaly value will be crucial to IoT services.

REFERENCES

[1] P. Yang *et al.*, "Lifelogging data validation model for Internet of Things enabled personalized healthcare," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 1, pp. 50–64, Jan. 2018.

[2] C. Yin, J. Xi, R. Sun, and J. Wang, "Location privacy protection based on differential privacy strategy for big data in industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3628–3636, Aug. 2018.

[3] J. Wang, Y. Gao, W. Liu, W. Wu, and S.-J. Lim, "An asynchronous clustering and mobile data gathering schema based on timer mechanism in wireless sensor networks," *Comput. Mater. Continua*, vol. 58, no. 3, pp. 711–725, 2019.

[4] J. Wang, Y. Gao, W. Liu, A. K. Sangaiah, and H.-J. Kim, "An improved routing schema with special clustering using PSO algorithm for heterogeneous wireless sensor network," *Sensors*, vol. 19, no. 3, pp. 671–687, 2019.

[5] C. Wang, Z. Zhao, L. Gong, L. Zhu, Z. Liu, and X. Cheng, "A distributed anomaly detection system for in-vehicle network using HTM," *IEEE Access*, vol. 6, pp. 9091–9098, 2018.

[6] C. Yin, L. Shi, R. Sun, and J. Wang, "Improved collaborative filtering recommendation algorithm based on differential privacy protection," *J. Supercomput.*, vol. 7, pp. 1–14, Jan. 2019.

[7] C. Yin, S. Ding, and J. Wang, "Mobile marketing recommendation method based on user location feedback," *Human Centric Comput. Inf. Sci.*, vol. 9, no. 1, pp. 1–14, 2019.

[8] J. Howcroft, J. Kofman, and E. D. Lemaire, "Prospective fall-risk prediction models for older adults based on wearable sensors," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 25, no. 10, pp. 1812–1820, Oct. 2017.

[9] X. Wang, "Intelligent multi-camera video surveillance: A review," *Pattern Recognit. Lett.*, vol. 34, no. 1, pp. 3–19, 2013.

[10] L. Yu, N. Wang, and X. Meng, "Real-time forest fire detection with wireless sensor networks," in *Proc. Int. Conf. Wireless Commun. Netw. Mobile Comput.*, vol. 2, 2005, pp. 1214–1217.

[11] Y. He, Y. Zhang, R. Kuligowski, R. Cifelli, and D. Kitzmiller, "Incorporating satellite precipitation estimates into a radar-gauge multisensor precipitation estimation algorithm," *Remote Sensors*, vol. 10, no. 1, pp. 106–125, 2018.

[12] J. Li and G. Hong, "Survey on sensor network research," *J. Comput. Res. Develop.*, vol. 45, no. 1, pp. 1–15, 2008.

[13] H. Fei, F. Xiao, G.-H. Li, L.-J. Sun, and R.-C. Wang, "An anomaly detection method of wireless sensor network based on multi-modals data stream," *Chin. J. Comput.*, vol. 40, no. 8, pp. 1–14, 2017.

[14] F. Giannoni, M. Mancini, and F. Marinelli, "Anomaly detection models for IoT time series data," 2018. [Online]. Available: arXiv:1812.00890.

[15] V. Kindl, B. Skala, R. Pechanek, V. Kus, and J. Hornak, "Low-pass filter for HV partial discharge testing," *Sensors*, vol. 18, no. 2, pp. 482–495, 2018.

[16] T. Fuse and K. Kamiya, "Statistical anomaly detection in human dynamics monitoring using a hierarchical dirichlet process hidden Markov model," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 11, pp. 3083–3092, Nov. 2017.

[17] N. Ding, H. Gao, H. Bu, and H. Ma, "RADM: Real-time anomaly detection in multivariate time series based on Bayesian network," in *Proc. IEEE Int. Conf. Smart Internet Things (SmartIoT)*. Los Alamitos, CA, USA, 2018, pp. 129–134.

[18] D. Codetta-Raiteri and L. Portinale, "Dynamic Bayesian networks for fault detection, identification, and recovery in autonomous spacecraft," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 1, pp. 13–24, Jan. 2015.

[19] J. Zhu, Y. Wang, D. Zhou, and F. Gao, "Batch process modeling and monitoring with local outlier factor," *IEEE Trans. Control Syst. Technol.*, vol. 27, no. 4, pp. 1–14, Jul. 2019.

[20] Kurnianingsih, L. E. Nugroho, Widyawan, L. Lazuardi, and A. S. Prabuwono, "Detection of anomalous vital sign of elderly using hybrid *k*-means clustering and isolation forest," in *Proc. Int. Tech. IEEE Region Conf.*, 2018, pp. 913–918.

[21] Y. Tian, M. Mirzabagheri, S. M. H. Bamakan, H. Wang, and Q. Qiang, "Ramp loss one-class support vector machine; A robust and effective approach to anomaly detection problems," *Neurocomputing*, vol. 310, pp. 223–235, Oct. 2018.

[22] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "LSTM-based encoder-decoder for multi-sensor anomaly detection," in *Proc. Int. Conf. Mach. Learn. (ICML) Anomaly Detect. Workshop*, New York, NY, USA, 2016, pp. 1–5.

[23] Y. Park and I. D. Yun, "Comparison of RNN encoder-decoder models for anomaly detection," 2018. [Online]. Available: arXiv:1807.06576.

[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[25] P. Liu, Z. Zeng, and J. Wang, "Multiple mittag-leffler stability of fractional-order recurrent neural networks," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 8, pp. 2279–2288, Aug. 2017.

[26] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[27] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," 2018. [Online]. Available: arXiv:1803.01271.

[28] S. Oehmcke, O. Zielinski, and O. Kramer, "Input quality aware convolutional LSTM networks for virtual marine sensors," *Neurocomputing*, vol. 275, pp. 2603–2615, Jan. 2018.

[29] T.-Y. Kim and S.-B. Cho, "Web traffic anomaly detection using C-LSTM neural networks," *Expert Syst. Appl.*, vol. 106, pp. 66–76, Sep. 2018.

[30] C. Zhou, C. Sun, Z. Liu, and F. C. M. Lau, "A C-LSTM neural network for text classification," 2015. [Online]. Available: arXiv:1511.08630.

[31] P. Esling and C. Agon, "Time-series data mining," *ACM Comput. Surveys*, vol. 45, no. 1, pp. 1–34, 2012.

[32] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *Proc. 23rd Eur. Symp. Artif. Neural Netw.*, Bruges, Belgium, 2015, pp. 89–94.

[33] A. K. Bhunia, A. Konwer, A. K. Bhunia, A. Bhowmick, P. P. Roy, and U. Pal, "Script identification in natural scene image and video frames using an attention based convolutional-LSTM network," *Pattern Recognit.*, vol. 85, pp. 172–184, Jan. 2019.

[34] F. T. Dezaki *et al.*, "Cardiac phase detection in echocardiograms with densely gated recurrent neural networks and global extrema loss," *IEEE Trans. Med. Imag.*, vol. 38, no. 8, pp. 1–12, Aug. 2019.

[35] S. Zheng, J. Xu, P. Zhou, H. Bao, Z. Qi, and B. Xu, "A neural network framework for relation extraction: Learning entity semantic and relation pattern," *Knowl. Based Syst.*, vol. 114, pp. 12–23, Dec. 2016.

[36] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *J. Physiol.*, vol. 160, no. 1, pp. 106–154, 1962.

[37] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *Proc. IEEE Int. Conf. Intell. Security Informat. (ISI)*, 2017, pp. 43–48.

[38] C.-L. Liu, W.-H. Hsaio, and Y.-C. Tu, "Time series classification with multivariate convolutional neural network," *IEEE Trans. Ind. Electron.*, vol. 66, no. 6, pp. 4788–4797, Jun. 2019.

[39] F. Karim, S. Majumdar, H. Darabi, and S. Chen, "LSTM fully convolutional networks for time series classification," *IEEE Access*, vol. 6, pp. 1662–1669, 2018.

Authorized licensed use limited to: Southern University of Science and Technology. Downloaded on October 19,2023 at 13:17:59 UTC from IEEE Xplore. Restrictions apply.

[40] L. Wen, L. Gao, and X. Li, "A new deep transfer learning based on sparse auto-encoder for fault diagnosis," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 49, no. 1, pp. 136–144, Jan. 2019.
[41] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Montreal, QC, Canada, 2014, pp. 3104–3112.
[42] J. Takeuchi and K. Yamanishi, "A unifying framework for detecting outliers and change points from time series," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 4, pp. 482–492, Apr. 2006.
[43] L. Zhang, J. Lin, and R. Karim, "Sliding window-based fault detection from high-dimensional data streams," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 2, pp. 289–303, Feb. 2017.
[44] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. W. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, 2017, pp. 2627–2633.

Chunyong Yin received the B.S. degree in computer science from the Shandong University of Technology, Zibo, China, in 1998, and the M.S. and Ph.D. degrees in computer science from Guizhou University, Guiyang, China, in 2005 and 2008, respectively.

He was a Postdoctoral Research Associate with the University of New Brunswick, Fredericton, NB, Canada, in 2011 and 2012. He is currently a Professor and the Dean with the Nanjing University of Information Science and Technology, Nanjing, China. He has authored or coauthored more than twenty journal and conference papers. His current research interests include privacy preserving and sensor networking, machine learning, and network security.

Sun Zhang received the bachelor's and master's degrees in computer science and technology from the Nanjing University of Information Science and Technology, Nanjing, China, in 2015 and 2018, respectively, where he is currently pursuing the Ph.D. degree in computer science.

His current research interests are deep learning and network security.

Jin Wang (Member, IEEE) received the B.S. and M.S. degrees in information engineering from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 2002 and 2005, respectively, and the Ph.D. degree in computer science from Kyung Hee University Korea, Seoul, South Korea, in 2010.

He is currently a Professor with the School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha, China. His research interests mainly include routing algorithm design, performance evaluation and optimization for wireless ad hoc and sensor networks.

Prof. Wang is a member of ACM.

Neal N. Xiong (Senior Member, IEEE) received the first Ph.D. degree in sensor system engineering from Wuhan University, Wuhan, China, in 2008, and the second Ph.D. degree in dependable sensor networks from the Japan Advanced Institute of Science and Technology, Nomi, Japan, in 2012.

Before he attended Northeastern State University, he worked with Georgia State University, Wentworth Technology Institution, and Colorado Technical University (full professor about 5 years) about 10 years. He is currently an Associate Professor with the Department of Mathematics and Computer Science, Northeastern State University, Tahlequah, OK, USA, for five years. His research interests include cloud computing, security and dependability, parallel and distributed computing, networks, and optimization theory.

Dr. Xiong is a Senior Member of IEEE Computer Society.

Authorized licensed use limited to: Southern University of Science and Technology. Downloaded on October 19,2023 at 13:17:59 UTC from IEEE Xplore. Restrictions apply.