TSMAE: A Novel Anomaly Detection Approach for Internet of Things Time Series Data Using Memory-Augmented Autoencoder

Honghao Gao, *Senior Member, IEEE*, Binyang Qiu, Ramón J. Durán Barroso, Walayat Hussain, Yueshen Xu, *Member, IEEE*, and Xinheng Wang, *Senior Member, IEEE*

***Abstract*—With the development of communication, the Internet of Things (IoT) has been widely deployed and used in industrial manufacturing, intelligent transportation, and healthcare systems. The time-series feature of the IoT increases the data density and the data dimension, where anomaly detection is important to ensure hardware and software security. However, for the general anomaly detection methods, the anomaly may be well-reconstructed with tiny differences that are hard to discover. Measuring model complexity and the dataset feature space is a long and inefficient process. In this paper, we propose a memory-augmented autoencoder approach for detecting anomalies in IoT data, which is unsupervised, end-to-end, and not easily overgeneralized. First, a memory mechanism is introduced to suppress the generalization ability of the model, and a memory-augmented time-series autoencoder (TSMAE) is designed. Each memory item is encoded and recombined according to the similarity with the latent representation. Then, the new representation is decoded to generate the reconstructed sample, based on which the anomaly score can be obtained. Second, the addressing vector tends to be sparse by adding penalties and rectification functions to the loss. Memory modules are encouraged to extract typical normal patterns, thus inhibiting model generalization ability. Long short-term memory (LSTM) is introduced for decoding and encoding time-series data to obtain the contextual characteristics of time-series data. Finally, through experiments on the ECG and Wafer datasets, the validity of the TSMAE is verified. The rationality of the hyperparameter setting is discussed by visualizing the memory module addressing vector.**

***Index Terms*—Anomaly detection, time-series classification, memory augmentation, autoencoder.**

Manuscript received 1 November 2021; revised 14 March 2022; accepted 26 March 2022. Date of publication 29 March 2022; date of current version 20 September 2023. This work was supported by the National Key Research and Development Program of China under Grant 2020YFB1006003 Recommended for acceptance by Dr. Mainak Adhikari. *(Corresponding author: Honghao Gao.)*

Honghao Gao and Binyang Qiu are with the School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China (e-mail: gaohonghao@shu.edu.cn; qby98@shu.edu.cn).

Ramón J. Durán Barroso is with the Faculty of Telecommunication Engineering, University of Valladolid, 47002 Valladolid, Spain (e-mail: rduran@tel.uva.es).

Walayat Hussain is with Victoria University Business School, Victoria University, Melbourne, VIC 3011, Australia, and also with the University of Technology Sydney, Ultimo, NSW 2007, Australia (e-mail: walayat.hussain@vu.edu.au).

Yueshen Xu is with the School of Computer Science and Technology, Xidian University, Xi'an 710126, China (e-mail: ysxu@xidian.edu.cn).

Xinheng Wang is with the School of Advanced Technology, Xi'an Jiaotong-Liverpool University, Suzhou 215123, China (e-mail: Xinheng.Wang@xjtlu.edu.cn).

Digital Object Identifier 10.1109/TNSE.2022.3163144

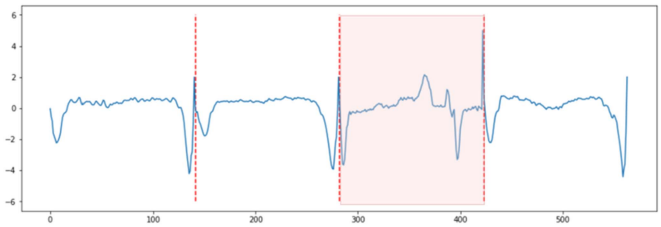


Fig. 1. For anomalous samples in ECG data, each segment divided by dashed lines is multidimensional heartbeat data generated by multiple sensors in one time unit. The data in the red interval are anomalous samples.

I. Introduction

THE Internet of Things (IoT) has been extensively studied and deployed in recent years [1], [2]. With a large number of seamlessly deployed sensors and controllers, simulation, prediction, and control of the entire production cycle can be achieved [3]. The sensor collects data at a certain frequency and sends it to the corresponding data receiver. The data received by the receiver, which are time-series data, are sorted and stored in strict chronological order. Contextual data points are highly correlated and causally related [4]. The IoT system contains a large number of data communication and transmission devices. Due to large-scale, computational, and storage complexity, data anomalies are unavoidable [5], [6]. Higher bandwidth and lower latency also extend the attack surface [7], [8]. Abnormal changes in the monitored environment or device anomalies produce significant differences in the collected data. These anomalies may pose a great risk to the security of the whole system [9]. Therefore, effective anomaly detection methods are needed to detect and identify the occurrence of anomalies.

Anomalies are samples that are significantly different from most other samples in terms of characteristics [10], as shown in Fig. 1. Each segment of the dashed line segmentation is the data of one heartbeat collected by one sensor, and each segment is a sample. The three segments from 0 to 140, 140 to 280, and 420 to 560 in the figure are normal samples. The third segment with a red marked area from 280 to 420 is an abnormal sample. There are obvious differences in the intermediate data segments between the abnormal samples and the other three normal samples. The anomaly detection classification task requires judging a small number of anomalous samples from a large number of normal samples, and usually, the

number of support samples for the category is extremely unbalanced, with a small number of anomalous samples and a low probability of occurrence. Today's devices in large IoT systems have high frame rates of data collected during each acquisition and large time sequence lengths included in the results as sensor technology are upgraded and data transmission capabilities are improved. This makes each data dimension high, and traditional machine learning algorithms cannot model them efficiently and accurately [11], [12]. The powerful expressiveness of deep learning models is good at modeling high-dimensional data while being able to extract contextual information features for time-series data. However, without exact model complexity and data dimensionality measurements, it is difficult to control the generalization ability of the model [13], which means that anomalies can sometimes be represented. Additionally, measuring model complexity and the dataset feature space is a long and inefficient process, so an unsupervised, end-to-end, expressive but not easy-to-generalize approach is needed. The contextual information of the temporal data is also very important and must be added to the sample reconstruction process [14], [15].

Density-based anomaly detection methods such as local outlier factor (LOF) [16] and density-based spatial clustering of applications with noise (DBSCAN) [17] calculate the density by the data distribution in each feature space. This method considers the cluster density in which the normal sample is located to be higher than the cluster density in which the anomalous sample is located as the judgment of the anomaly. However, it is inefficient in the high-dimensional case because density estimation is performed once in each dimension. The isolation-based anomaly detection method is typified by the isolation forest [18]. A random hyperplane is used to partition the data space, generating two subspaces. Another random hyperplane is then used to partition each subspace until there is only one sample in each subspace. Intuitively, cluster elements with high density need to be partitioned many times before they stop. By building a tree of this process, normal samples with high cluster density should generally be farther from the root node than abnormal samples, so the distance to the root node can be used as the criterion for determining abnormality. However, the isolation forest becomes less effective for the additional noise brought by high-dimensional data. A small number of samples cannot support the algorithm for a complete description, so anomalous patterns are generally excluded by learning the patterns of a large number of normal samples. One of the efficient and commonly used methods of anomaly detection algorithms is the autoencoder (AE) [19], [20]. It is reconstructed by unsupervised learning of the entire dataset, using the samples to train the model. It consists of an encoder and a decoder, where the encoder compresses the input into a vector of latent representations, and the decoder can reconstruct the data from it. The sample is reconstructed by the AE, using itself as the ground truth of the reconstruction result. This is essentially a form of information compression, forcing the model to record the most typical features of the sample in an encoding that uses the specified information capacity. In anomaly detection tasks, the reconstruction error is generally used as an indicator for determining anomalies. Normal data with a large proportion of samples will naturally have a much greater impact on the parameters than those with a small proportion of samples. It is generally assumed that the reconstruction error is low for normal data because they are close to the original sample, while it is high for anomalous data [21]. However, this assumption does not always hold in the anomaly detection task for time series. The AE sometimes generalizes well, making the anomalous data also well reconstructed. This results in anomalous samples also possessing a reconstruction error that is quite low compared to normal samples, which are then indistinguishable. This is because the learning ability of the AE is too strong and the parameter capacity is too large, which allows the model to learn the patterns of normal samples as well as the patterns of abnormal samples. This phenomenon of "overgeneralization" [22], [23] due to the strong generalization ability is common in time-series anomaly detection.

To overcome the drawbacks of the AE, this paper proposes adding a memory module to the deep AE and designing a time-series memory autoencoder (TSMAE) for time-series data anomaly detection. Given an input, it is encoded as a latent representation using long short-term memory (LSTM) in the TSMAE and then used as a query to retrieve the distance to each memory in memory. Then, the memories are weighted together according to the distances to produce the most similar latent representation of the sample with anomalies eliminated. Finally, the new latent representation is decoded to generate the reconstructed samples. Thus, model generalization is suppressed so that the reconstructed samples of abnormalities are also similar to normal samples, and the reconstruction error of abnormal samples is strengthened. The main contributions of this paper are as follows:

(1) An AE model with memory enhancement for IoT time-series anomaly detection is proposed. It has the advantages of unsupervised, end-to-end, and expressive capabilities but is not easy to overgeneralize compared to traditional methods. The model uses LSTM for encoding and decoding and has better feature extraction capability for time series.

(2) Two constraints are designed in the loss function to encourage the model to extract patterns using fewer memory terms with better-reconstructed samples. Combining the rectification operator eliminates the low-weighted miscellaneous terms to remove unnecessary robustness. This further suppresses the generalization ability of the model.

(3) Experiments on two public datasets demonstrate the effectiveness of TSMAE, outperforming the results of other baseline algorithms. The hyperparameters are also adjusted to visualize the change process of the addressing vector, and the principle of the TSMAE to suppress generalization is demonstrated.

This paper is organized as follows. Section II shows the problems of anomaly detection tasks for temporal data in real-world scenarios. Section III reviews related work in deep learning anomaly detection and memory enhancement models for temporal data. Section IV describes the specific design of the TSMAE. Section V demonstrates the effectiveness of the

TSMAE through several experiments and visually shows the TSMAE suppression generalization process.

II. Motivation

There are two common approaches to the anomaly detection task, which produce discrete and continuous results for detection. The discrete approach gives a direct determination of whether the sample is anomalous. A piece of temporal data collected $\mathbf{x} = \{x_1, x_2, \ldots, x_t\} \in \mathbb{R}^T, 1 \leq t \leq T$, is taken as input. The detection model is denoted by C. The detection result is $Y \in \{-1, 1\}$, $Y = 1$ for abnormal samples and $Y = -1$ for normal samples. For any column vector of $\mathbf{x}$, the contextual time dependence is unknown. Detection using C for $\mathbf{x}$ is a time-series classification problem, which can be abstracted as finding model C such that it satisfies

$$C(\mathbf{x}, g(\cdot)) = Y \tag{1}$$

Model C establishes the classification hyperplane $g(\cdot)$ in each feature space by learning so that samples brought into the hyperplane equation can be directly judged

$$Y = \begin{cases} -1, & g(\boldsymbol{x}) < 0 \\ 1, & g(\boldsymbol{x}) \geq 0. \end{cases} \tag{2}$$

However, in practice, the number of anomalous samples is much smaller than that of normal samples. This makes the classification hyperplane $g(\cdot)$ not describe the anomalies well and prefers to classify them as normal or even ignore the anomalies completely. Therefore, the continuous method is widely used in industry. In the IoT time sequence data scenario, the attributes are usually continuous data, and it is easy to obtain continuous judgment indicators. Therefore, this method is very common in this scenario. That is, model C gives the anomalous value of the sample, and the classification detection is performed by comparison with a preset threshold [24]. With φ denoting the anomalous value of the sample and τ denoting the set threshold, the problem is abstracted to find model C such that

$$C(\mathbf{x}, f(\cdot)) = \varphi \tag{3}$$

where φ can be a dimensionless real number or a probability value between 0 and 1. The judgment for the abnormal result then becomes

$$Y = \begin{cases} -1, & \varphi < \tau \\ 1, & \varphi \geq \tau. \end{cases} \tag{4}$$

The AE uses Eqs. (3) and (4) in the anomaly detection task. For a temporal data sample $\mathbf{x} = \{x_1, x_2, \ldots, x_t\}$, $x_t \in \mathbb{R}$ of length T, $1 \leq t \leq T$. The AE maps the sample to the feature space to obtain a latent representation vector of length E, $\mathbf{z} \in \mathbb{R}^E$, and expects the latent representation to reconstruct the original sample. Let $f_e(\cdot) : \mathbb{R}^T \to \mathbb{R}^E$ denote the encoder and $f_d(\cdot) : \mathbb{R}^E \to \mathbb{R}^T$ denote the decoder. The AE reconstruction process [25] is

$$\mathbf{z} = f_e(\mathbf{x}) \tag{5}$$

$$\hat{\mathbf{x}} = f_d(\mathbf{z}) \tag{6}$$

The L2 distance between $\mathbf{x}$ and $\hat{\mathbf{x}}$ is taken as the anomaly score φ

$$\varphi = \|\mathbf{x} - \hat{\mathbf{x}}\|_2 \tag{7}$$

The AE-based anomaly detection method exploits the nature of the detection task where the number of anomalous samples is extremely low. This makes the AE more inclined to learn the reconstruction of normal samples by having the parameters corrected by a larger number of normal samples during the training process. This makes the reconstruction data of the abnormal samples similar to the normal samples, so it can make the abnormal fraction φ of the normal samples smaller. The anomalous fraction φ of anomalous samples is larger, thus distinguishing the two kinds of samples. However, in the large quantity of temporal data generated in the IoT scenario, the AE often overgeneralizes, making the AE also learn to reconstruct the anomalous samples well. As shown in Fig. 2, the experiments conducted on the Wafer dataset are shown in this paper. (a) shows the normal sample and its reconstruction result. The AE fits the pattern of the original sample, which makes the error between the original and the reconstructed sample low, and (b) shows the abnormal sample and its reconstruction result, in which the AE still fits well.

In the case where the number of abnormalities is very small relative to the number of normal samples, the AE trains on the ability to reconstruct samples to learn a large number of normal samples. Thus, reconstructing abnormal samples can also produce samples that resemble normal samples, resulting in a large difference. There is more than one type of normal sample and some differences between them. Therefore, the model complexity increases so that the robustness and generalization improve. This leads to better normal sample reconstruction. However, in practice, due to the overexpression ability of the model, normal samples are well reconstructed, while the abnormal sample reconstruction ability is also learned in a balanced way. It is usually desired that the generalization ability of the model can enhance the robustness of the normal samples without learning the reconstruction ability of the abnormal samples. Therefore, when this phenomenon occurs, we think it is more appropriate to call it overgeneralization, which is essentially an overfitting phenomenon. However, during the design process, it takes many trials to accurately measure model complexity to fit the data space in the task, which consumes considerable time and computational resources. Therefore, an unsupervised, end-to-end, and not easy-to-generalize approach is needed.

Based on this, a memory module is added to Eqs. (5) and (6) to match the latent representation of the AE mapped to the feature space with the latent representation of normal samples in the memory module. Recombining a latent representation of the most similar sample eliminates the anomaly and then decodes this latent representation to generate the reconstructed sample with the original sample to calculate the outlier. This scheme does not change the model complexity by adjusting the number of parameters but rather blocks the propagation of anomalies to suppress the model generalization ability. It enables us to quickly find models that do not overgeneralize by

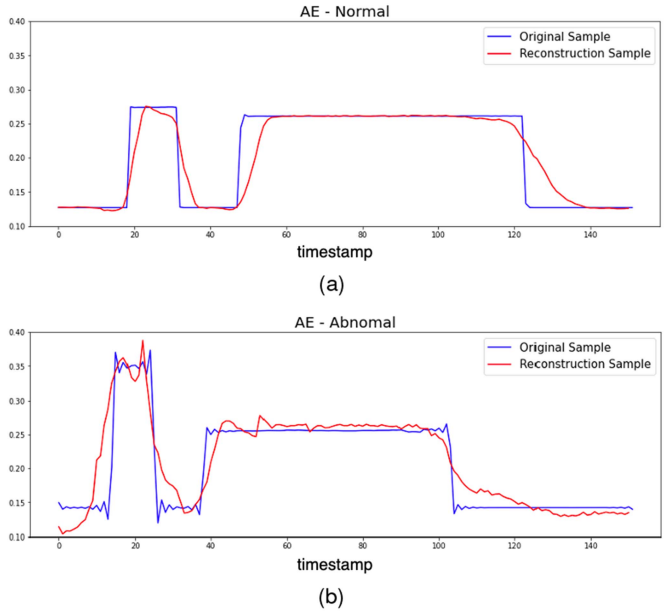


Fig. 2. For the observation of AE abnormal sample reconstruction, the data are from the Wafer dataset, and the reconstruction results are in red. (a) Normal sample and its reconstruction results (b) Abnormal sample and its reconstruction results

unsupervised, end-to-end efficient training in the rapidly changing environmental data of IoTs. Additionally, considering that the temporal information of data in IoT scenarios is important, LSTM is introduced for decoding and encoding work to obtain the contextual features of temporal data [26].

III. Related Work

Due to the sparse number of anomalies, it is difficult to accurately describe anomaly samples based on a limited number of anomalies. There are so many anomalies that it is impossible to list all the anomalous possibilities. Therefore, unsupervised methods are more often used for deep learning. For anomaly detection scenarios of time-series data, there are two main solution types. One type generates predicted data based on historical data and compares it with input samples to detect anomalies. Wu [27] used LSTM for predictive modeling of time-series data and then used a naive Bayesian model to evaluate the error to detect outliers. In contrast, Meng [28] used a transformer for time-series prediction, and the parallel computing capability of the transformer and the advantage of extracting relations across distances make the computational time shorter while improving the anomaly identification ability. Yin [29] introduced convolutional neural networks (CNNs) for feature extraction and used sliding windows to extract features to provide LSTM for prediction comparison, thus comparing anomalies. This solution type has high real-time performance but insufficient global observation, poor adaptability, and a greater impact from environmental changes.

Another type of model trained based on historical data reconstructs the input samples and compares the reconstruction results with the input samples to detect anomalies. Taking advantage of the fact that anomalies are in the minority in the dataset, the trained model can reconstruct normal samples, so the reconstruction error for abnormal samples is larger. Malhotra [30] proposed LSTM as an autoencoder model for encoders and decoders, using the contextual information extraction capability of LSTM to better reconstruct the time series and compare the reconstruction errors, but this approach does not consider spatiality. Zhang [31] added a CNN combined with LSTM to capture patterns in time and space and combined this network as an autoencoder architecture to determine anomalies from multiscale reconstruction comparisons. Li [32] proposed a smoothed induced sequence variational autoencoder (VAE) with a recurrent neural network (RNN) to implement the VAE backbone network, penalizing the non-smoothed reconstruction using prior knowledge of smoothing induction and using two criteria, reconstruction probability and reconstruction error, to determine the anomaly. The reconstruction-comparison approach reduces some real-time but places more emphasis on global contextual information and is more flexible to adapt to changes in the environment in unsupervised scenarios. The scheme adopted in this paper is the reconstruction-comparison approach, using LSTM as the AE backbone for reconstruction.

Memory networks were first proposed by Weston [33] and applied in the question-answering (QA) field, where the memory modules act as a dynamic knowledge base but require separate supervised training for each intermediate module during the training process. Sukhbaatar [34] proposed an end-to-end training approach that reduces coupling and creates more flexibility for the memory mechanism. It has much less supervision during training than the former and is more generally applicable in realistic settings. Han [35] used the memory module to compress the patterns of multiple actions in memory, and multiple memory items were combined to provide the model to make multiple assumptions about the unknown actions. Milde [36] embedded the memory module into a codec network composed of LSTMs and combined the records generated from memory with the contextual features extracted by the LSTM for decision making. Chang [37] used the memory module as a dynamic knowledge base to capture its long-term patterns. Additionally, it is easy to know which part of the historical data is most referenced using the properties of the memory module, making the model highly interpretable.

The memory module, as a plug-and-play dynamic knowledge base, possesses properties such as high interpretability, memorizing multiple typical patterns, and providing auxiliary information for long-term memory. By removing the last step of the original Sukhbaatar's end-to-end model [34] and the combination of the original potential representation, only the other two memory module properties are used. It is possible to obtain a module that generates information only by combining typical patterns in historical memory items, which is excellent for blocking the propagation of subtle features. It has a good ability to suppress generalization. In the method that uses reconstruction error to determine anomalies, adding the memory module blocks the transmission of anomalous information. Combining potential representations using only the patterns of normal samples in forward propagation makes the reconstruction error of

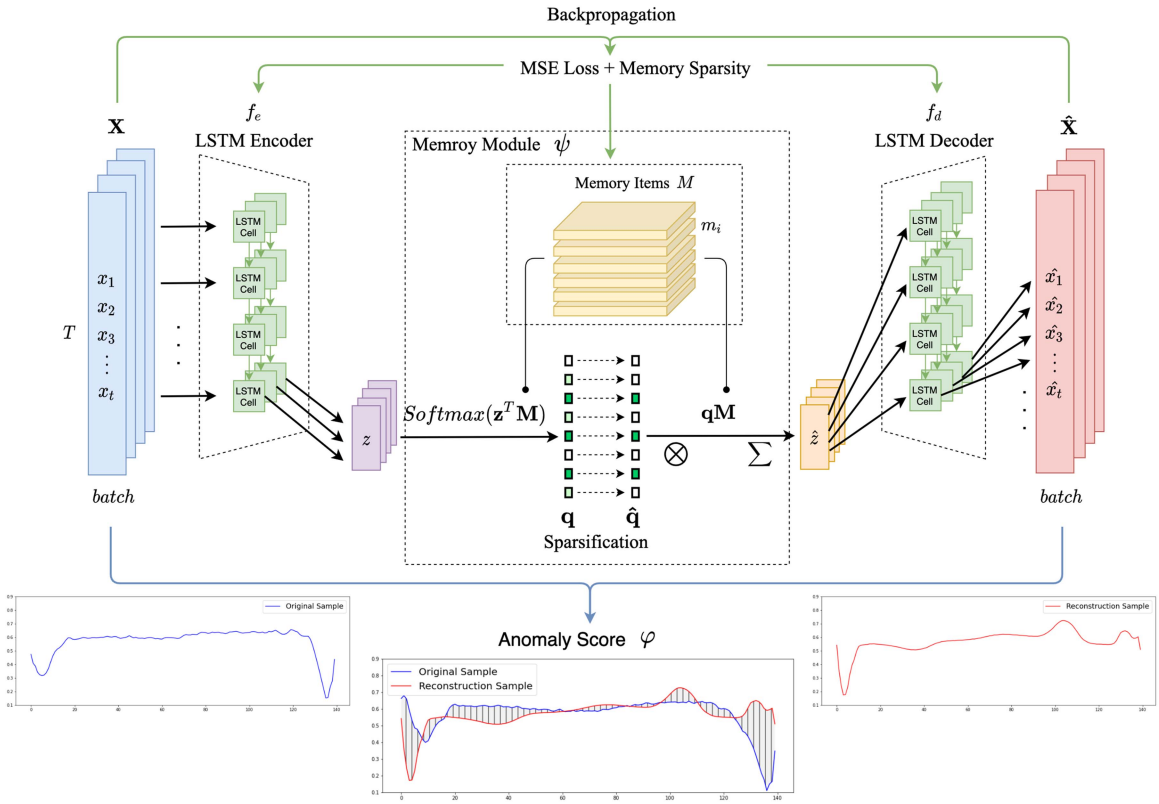


Fig. 3. An overview of the TSMAE.

anomalous samples high. In turn, the attention addressing the combined memory term provides robustness for multiple patterns of normal samples. It forces a larger reconstruction error between anomalous and normal samples and provides the ability to not overgeneralize the rapidly changing IoT time-series data. LSTM is also introduced in the before-and-after codec of the memory module to extract temporal information.

IV. Proposed Model

A. Time-Series Data Preprocessing

Raw IoT sensors vary widely in the range of attribute values of the data collected in different situations. Data with larger values tend to prop up the scale interval, leading to insensitive change models for smaller values and thus ignoring their attribute characteristics. Therefore, raw data need to be normalized from the entire time dimension based on the contextual data of the sensor at the same point in time each time.

The complete raw data $\mathbf{D} \in \mathbb{R}^{n \times T}$ is a matrix containing n time-series samples. Each sample is of length T, which denotes the data collected by T acquisition actions. For each of these data points d, the maximum and minimum values of the column in which the data are located are found as the normalization parameters for that data point. The normalized data x are obtained as in Eq. (8).

$$x = \frac{d - ColMin(d, \mathbf{D})}{ColMax(d, \mathbf{D}) - ColMin(d, \mathbf{D})} \tag{8}$$

B. Model Architecture

The TSMAE proposed in this paper consists of three main parts: an encoder, a memory module, and a decoder. As shown in Fig. 3, the bold black solid line shows the inference process of forward propagation. The arrows indicate the direction of data flow. Given an input, the encoding of the input is first obtained by the encoder. With the encoding as a query, the memory module finds the direct distance to each memory item in the memory and then combines it into a new encoding based on the distance to pass to the decoder for reconstruction. The solid blue line indicates that an error value is measured as the anomaly score by comparing the reconstruction result with the original sample. The anomaly is determined by the size of the anomaly score. The added memory module blocks the transmission of anomalous information, and the latent representation used for decoding is not the result of the feature-compressed representation of the original samples but the result of combining the memory items in the memory module with similarity as the weight. Due to the update of a large number of normal samples, the memory items are increasingly represented as the compression of the normal pattern, while the latent representation of the abnormal samples can only be obtained by the combination of the latent representation memory items of the normal samples, forcibly removing the abnormal information and making the reconstruction result similar to the normal pattern. The parameters of the three parts are initially obtained by random initialization and then adjusted by backpropagation during the training process. The green solid

line indicates the backpropagation to adjust the parameters of the decoder, encoder, and memory module.

Distinguishing from the basic AE structure in Eqs. (5) and (6), the TSMAE adds the step of Eq. (9). The encoder $f_e(\cdot)$ compresses $\mathbf{x}$ into a latent representation $\mathbf{z}$, and $\mathbf{z}$ obtains the reconstructed latent representation $\hat{\mathbf{z}}$ through the memory module $\boldsymbol{\psi}$ and then trains the decoder $f_d(\cdot)$ to decode the reconstructed $\hat{\mathbf{z}}$ into the reconstructed sample $\hat{\mathbf{x}}$.

$$\hat{\mathbf{z}} = \Psi(\mathbf{z}) \tag{9}$$

For the vanilla AE model, $\hat{\mathbf{z}} = \mathbf{z}$, while in the TSMAE, $\hat{\mathbf{z}}$ is reconstructed by addressing the combination in the memory module ψ with $\mathbf{z}$ as a query, and the latent representation of the reconstruction is $\hat{\mathbf{z}} \in \mathbb{R}^E$.

The temporal relationship before and after each time-series data item needs to be considered. Therefore, TSMAE uses LSTM as an encoder to generate the query for addressing memory items and uses sigmoid as the activation function of the hidden layer [25]. LSTM is a special kind of RNN (recurrent neural network) that consists of multiple consecutive cells. Unlike RNN, each cell uses $\mathbf{c}_t \in \mathbb{R}^H$ and $\mathbf{h}_t \in \mathbb{R}^H$ to record the intermediate states. $\widetilde{\mathbf{c}}_t \in \mathbb{R}^H$ is the candidate state obtained by the nonlinear function. The forget gate $\mathbf{f}_t \in [0,1]^H$ is added to control the discard of $\mathbf{c}_t$, the input gate $\mathbf{i}_t \in [0,1]^H$ controls the update of candidate state $\widetilde{\mathbf{c}}_t$ on $\mathbf{c}_t$, and the output gate $\mathbf{o}_t \in [0,1]^H$ controls the output of internal state $\mathbf{c}_t$ to external state $\mathbf{h}_t$ [38]. The computational process of each LSTM cell can be expressed as:

$$\begin{bmatrix} \widetilde{\mathbf{c}}_t \\ \mathbf{o}_t \\ \mathbf{i}_t \\ \mathbf{f}_t \end{bmatrix} = \begin{bmatrix} \tanh \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} \left(\mathbf{W} \begin{bmatrix} \mathbf{x}_t \\ \mathbf{h}_{t-1} \end{bmatrix} + \mathbf{b} \right) \tag{10}$$

$$\mathbf{c}_t = \mathbf{f}_t * \mathbf{c}_{t-1} + \mathbf{i}_t * \widetilde{\mathbf{c}}_t \tag{11}$$

$$\mathbf{h}_t = \mathbf{o}_t * \tanh(\mathbf{c}_t) \tag{12}$$

$$\sigma(x) = \mathrm{Sigmoid}(x) = \frac{1}{1+e^{-x}} \tag{13}$$

where $\mathbf{x}_t \in \mathbb{R}^T$ is the input at the current moment, $\mathbf{W} \in \mathbb{R}^{4H\times(H+T)}$ and $\mathbf{b} \in \mathbb{R}^{4H}$ are the learnable network parameters, and $*$ is the Hadamard product. Combining the output states of each cell yields the encoded latent representation $\mathbf{z}$.

$$\mathbf{z} = \{\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_T\} \tag{14}$$

Finally, $\mathbf{z}$ is fed into the memory module to eliminate anomalous features by addressing combinations.

C. Memory Module

The encoder produces the latent representation $\mathbf{z} \in \mathbb{R}^E$. Define $\mathbf{M} \in \mathbb{R}^{N\times E}$ as the memory module consisting of N memory items $\mathbf{m} \in \mathbb{R}^E$, and $\mathbf{m_i}$ denotes the i-th memory item. The attention addressing vector $\mathbf{q} \in \mathbb{R}^N$ is obtained by calculating the distance between the latent representation and each memory item.

$$q_i = Softmax(\mathbf{z}^T \mathbf{m}_i) \tag{15}$$

where $Softmax(z_i) = e^{z_i} / \sum_j e^{z_j}$ is used here. The matching degree between the latent representation and each memory item is calculated using the inner product, and then a relative similarity distance is obtained by comparing softmax with other memory items. The similarity distance with each memory item constitutes the addressing vector.

The items with high similarity represent the more likely normal patterns of the samples, so the memory items with high similarity in q_i should be more involved in describing the reconstructed latent representation $\hat{\mathbf{z}}$. Let addressing vector q_i be multiplied and summed up as scale factors of the memory items to obtain the reconstructed latent representation $\hat{\mathbf{z}}$.

$$\hat{\mathbf{z}} = \mathbf{q}\mathbf{M} = \sum_{i=1}^{N} q_i \mathbf{m}_i \tag{16}$$

The dense addressing vector $\mathbf{q}$ can still reconstruct anomalies by complex combinations of multiple memory items. To avoid this situation from arising, the rectified addressing vector technique of paper [39] is adopted in Eq. (17). As a weight factor, the addressing vector must have a sum of 1 for the final composite memory item to be comparable, so a standardization step in Eq. (18) needs to be added to make the sum of the addressing vector equal to 1.

$$q_i = \frac{\max(q_i - \lambda, 0) \cdot q_i}{|q_i - \lambda|} \tag{17}$$

$$\mathbf{q} = \frac{\mathbf{q}}{\max(\|\mathbf{q}\|_1, \varepsilon)} \tag{18}$$

where λ is an adjustable sparsification threshold. The sparse addressing vector uses a convenient backpropagation calculation to set the items with similarity below the threshold λ to 0. This forces the model to use fewer but more relevant memory items to compose the latent representation, thus learning more informative representations in a single memory item.

The setting of the sparsity threshold λ is related to the size N of the memory module. On average, each memory item has $\frac{1}{N}$ probability of participating in the composition of the latent representation $\hat{\mathbf{c}}$. To make the sparse addressing vector as sparse as possible, the sparsification threshold should satisfy at least

$$\lambda \geq \frac{1}{N} \tag{19}$$

The computational flow of the TSMAE is shown in Algorithm 1. First, the raw data are preprocessed to obtain computable temporal data $\mathbf{x}$. Then, the encoder $\mathbf{f_e}$ is used to encode $\mathbf{x}$ into a latent representation $\mathbf{z}$. Then, softmax is used to calculate the distance between the latent representation and each memory item to form the addressing vector $\mathbf{q}$. And it is rectified and normalized to make the addressing combination more efficient. The new latent representation $\hat{\mathbf{z}}$ is created by combining the memory items $\mathbf{M}$ with the weights of the addressing vector. The reconstructed sample $\hat{\mathbf{x}}$ is then obtained by decoding using a decoder. Then, the reconstruction error between the original sample $\hat{\mathbf{x}}$ and the reconstructed sample $\hat{\mathbf{x}}$ is calculated by MSE as the anomaly score φ. Finally, according to the data

Algorithm 1: Progress of the TSMAE.

Input: Time-series data $\mathbf{x}$, memory size N, sparsification threshold λ
Output: Anomaly score φ

```
Initialize M
z = f_e(x) // Encode
for m_i in M do
    q_i = z^T m_i / Σ_j z^T m_j // Calculate addressing vector by softmax
    q_i = max(q_i − λ, 0)·q_i / |q_i − λ| // Rectified q
end for
q = q / max(||q||_1, ε) // Normalize
ẑ = qM = Σ_{i=1}^{N} q_i m_i // Combine memory items
x̂ = f_d(z) // Decode
φ = ||x − x̂||_2 // Anomaly score by MSE
```

distribution of the anomaly score, the optimal threshold is set as the criterion to judge the normal and anomalous results in the task.

D. Loss Function

During the training process, normal and abnormal samples are trained together in the unlabeled case. The model is forced to extract typical features in the memory items by supervising how well the model reconstructs the samples, so minimizing the reconstruction error $\mathcal{L}_{rec}$ is needed as one of the learning objectives. TSMAE uses the mean square error to measure $\mathcal{L}_{rec}$.

$$\begin{aligned}\min \mathcal{L}_{rec} &= \frac{1}{2n} \min_{\theta_{f_e}\theta_{f_d}\theta_M} \|\mathbf{x} - \widehat{\mathbf{x}}\|_2 \\ &= \frac{1}{2n} \min_{\theta_{f_e}\theta_{f_d}\theta_M} \|\mathbf{x} - f_d(f_e(\mathbf{x}))\|_2\end{aligned} \tag{20}$$

θ_{f_e} and θ_{f_d} are the learnable parameters of the encoder and decoder. Another learnable parameter is the memory module parameter $\mathbf{M}$. The sparse addressing vector should be as sparse as possible, not only to avoid complex combinations so that anomalous samples can be well reconstructed but also to have each memory item remember typical normal patterns. Therefore, the sparse addressing vector cannot be compressed to zero by forcing the value to zero alone but also needs it to tend to be sparse. Therefore, $\mathcal{L}_{sparse}$ is designed to be a sparse metric as a penalty [40].

$$\mathcal{L}_{\text{sparse}} = \sum_{i=1}^{N} -\log\left(1 + q_i^2\right) \tag{21}$$

Thus, the final composition of the loss is

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \eta \mathcal{L}_{\text{sparse}} \tag{22}$$

The sparsity factor η is an adjustable hyperparameter. It is necessary to control the sparse addressing vector. The number of typical normal patterns varies from dataset to dataset depending on the differences in temporal data patterns. Insufficient sparsity can lead to failure of the sparsification operation, raising the problem that abnormal samples can also be well reconstructed. In addition, too much sparsity can lead to a reduction in the effective capacity of the model. That is, too much feature masking causes the model to fail to learn the effective features, making the anomaly scores of normal samples high as well.

TABLE I
DATASETS DETAILS

Dataset	Anomaly Ratio	Length	Size
ECG5000	8.75%	140	321
Wafer	9.70%	152	1,000
Arrhythmia	45.67%	274	452

V. EXPERIMENTS

A. Datasets

In this paper, three classical datasets in the field of time-series classification, ECG5000 [41] and Wafer [42], are used (Table I). ECG5000 is processed heartbeat detection data with 5 heartbeat types. The sample size of category 1 accounts for 58.4%, and the remaining categories 2-5 account for a total of 41.6%. Categories 2-5 have similar heartbeat types, while all are significantly different from category 1. Therefore, in this paper, some data from categories 2-5 are randomly sampled as anomalous data labeled as 1 and form a new data set with category 1. Wafer data are related to semiconductor manufacturing. The collection of measurements recorded online from various sensors during the processing of silicon wafers used in semiconductor manufacturing constitutes the Wafer database, and each data point in the database is a measurement recorded by a sensor during the processing of a wafer using a tool. There are two types of data, normal and abnormal, and a large quantitative imbalance between normal and abnormal data. The Arrhythmia [43] dataset aims to distinguish between the presence and absence of cardiac arrhythmia and to classify it into one of the 16 groups. Class 01 refers to normal ECG classes, and 02 to 16 refers to different classes of arrhythmia and unknown cases, accounting for 45.67%.

B. Baselines

Isolation forest [18] is a feature isolation tree-based anomaly detection algorithm. The dataset is continuously partitioned into multiple trees by randomly selecting features and randomly selecting split point locations. The algorithm considers that the shorter the average path length elapsed to the root node, the higher the possibility of being an anomaly. With linear time complexity and high accuracy, it has become a classical anomaly detection algorithm.

OC-SVM (one-class support vector machine) [44] is an unsupervised extension of SVM. OC-SVM needs to learn only one class of data. The algorithm solves an optimal hyperplane in the feature space so that as many normal data as possible fall within the decision boundary to obtain a minimum decision hyperplane. Those inside the hyperplane boundary are considered normal samples, and those outside are considered abnormal samples.

Vanilla AE (vanilla autoencoder) is a model of the most basic AE structure composed entirely of fully connected layers. Similar to the idea of PCA, vanilla AE compresses the sample into a segment of latent representation, restores the representation and forces the model to extract typical features recorded into the latent representation. Anomalies are judged by the error magnitude between the reconstructed sample and the original sample.

VAE (variational autoencoder) [45] is also a model of AE structure, but unlike vanilla AE, the encoder of VAE outputs the variational probability distribution of the latent representation, and then the new latent representation obtained by sampling from the distribution is decoded to reconstruct the sample. The anomaly is also determined by calculating the error between the reconstructed sample and the original sample.

OC-DeepSVDD (one-class deep support vector data description) [46] is based on the idea of OC-SVM, which uses neural network training to minimally compute the division hypersphere of the sample feature space. Then, it determines whether a sample point is anomalous based on the distance between the center of the sphere and the test sample point.

SO_GAAL (single-objective generative adversarial active learning) [47] uses the idea of GAN to generate potential outlier points (anomalous data) with a generator and combines noise and real data. Then, the discriminator discriminates between noise and real data and finally uses the discriminator as an anomaly detection classifier.

C. Metrics

The receiver operating characteristic (ROC) curve is a commonly used tool to visualize the classification performance of a binary classification model. The area under the ROC curve (AUC) is usually used to measure the classification performance of anomaly detection models. In the binary classification problem, TP (true positive) is the correctly detected anomaly, FP (false positive) is the incorrectly detected anomaly, and using P to denote all anomalous samples and N to denote all normal samples, the TPR and FPR can be calculated as

$$TPR = \frac{TP}{P} \tag{23}$$

$$FPR = \frac{FP}{N} \tag{24}$$

Different values of TPR and FPR are obtained for different thresholds τ. The ROC curve can be plotted by traversing all thresholds τ and using the obtained FPR as the horizontal axis and TPR as the vertical axis. The sum of the area under the curve is the AUC.

$$AUC = \int_0^1 TPR_{FPR} d(FPR) \tag{25}$$

D. Experiment Settings

The TSMAE in this paper uses a single-layer LSTM encoder in the experiments, the hidden unit is 10 and is combined with three fully connected layers for decoding. The number of hidden units is the same as the length of the sequence, and two dropout layers are added between the three layers, with the probability of random discard set to 20%. In the baseline model used for comparison, vanilla AE consists entirely of five fully connected layers with hidden units of $\{128, 32, 10, 32, 128\}$. The VAE encoder and decoder are both fully connected layers, compressed by the fully connected layer with a hidden unit of 100 into an implicit variable of length 10. The memory module sizes N for the ECG5000 and Wafer datasets are set to 20 and 100, respectively. $\mathcal{L}_{sparse}$ has a sparsity factor of 0.01 and 0.001. Arrhythmia and Wafer are the same. Sections 5.5.3 and 5.5.4 further discuss the effects of these two hyperparameters. The parameters of the memory module are initialized using the Xavier method [48].

TABLE II
AUC RESULTS FOR BASELINES AND TSMAE

Algorithm	ECG5000	Wafer	Arrhythmia
Isolation Forest	0.6128	0.9147	0.5301
OC-SVM	0.7740	0.7763	0.7498
VAE	0.7692	0.9135	0.7472
Vanilla AE	0.5395	0.9721	0.7511
OC-DeepSVDD	0.6324	0.6761	0.6089
SO_GAAL	0.8831	0.7395	0.6569
TSMAE	**0.9516**	**0.9912**	**0.7513**

E. Performance Evaluation

1) Baseline Comparison: Table II shows the AUC results of the baseline algorithm and the TSMAE, containing the results of the ablation experiments of the TSMAE on the memory module and LSTM. The results of the baselines on the three datasets are not as excellent as those of the TSMAE. For Wafer data with a low percentage and a high number of anomalies, the TSMAE achieves an excellent 0.9912. For ECG5000 with limited training samples, it also achieves 0.9516. On Arrhythmia, which has a very high percentage of anomalies, it also has better stability compared to other baseline algorithms, and the memory module does not retain more anomalous memories due to the high percentage of anomalies. Therefore, the TSMAE has a high tolerance for the anomaly ratio. However, in practice, we do not recommend using anomaly detection methods in such datasets, which no longer meet the characteristics of a "low percentage of anomalies" and are not a type of anomaly detection task.

Unlike the previous fittings in Fig. 2, the TSMAE fits the ECG5000 and Wafer datasets are shown in Fig. 4 below. The normal and abnormal samples differ mainly between 120 and 140. The red color shows the reconstruction results of the TSMAE, while the reconstruction results are similar for normal and abnormal samples and are close to normal samples between 120 and 140 and obviously not close to the pattern of abnormal samples, while effectively learning the pattern of normal samples. This makes the results of the abnormal

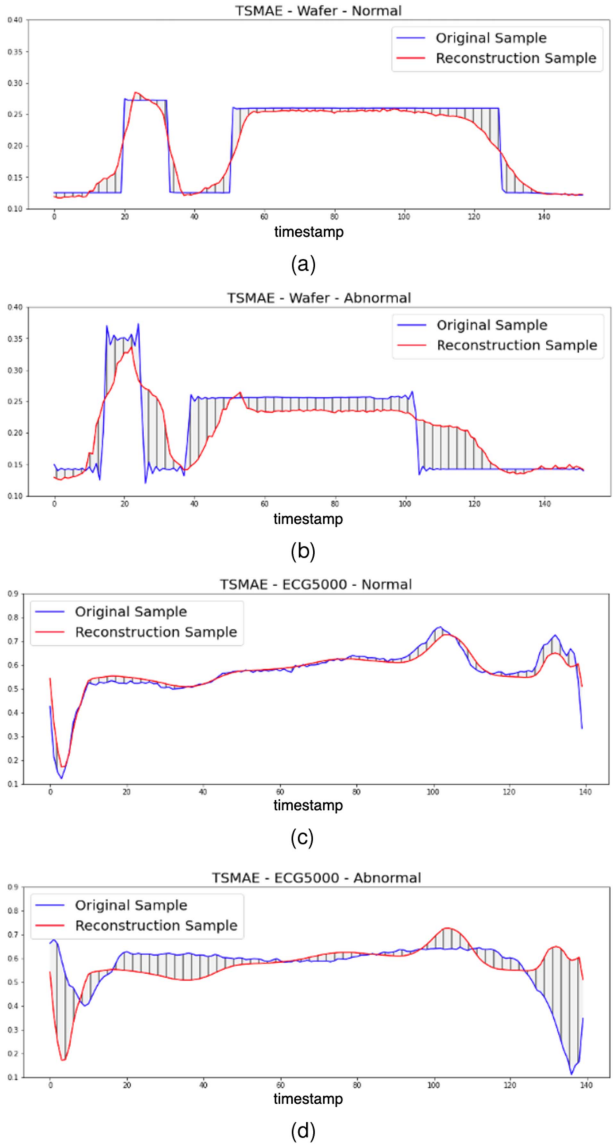


Fig. 4. To observe abnormal TSMAE reconstructed samples, the reconstruction results are shown in red. (a) Reconstruction results of TSMAE on the Wafer dataset for normal and abnormal samples. (b) Reconstruction results of TSMAE on the ECG5000 dataset for normal and abnormal samples. (a) Wafer Normal Sample (b) Wafer Abnormal Sample (c) ECG5000 Normal Sample (d) ECG5000 Abnormal Sample.

reconstruction also close to the normal sample, and the reconstruction error of the abnormal sample will be significantly higher than that of the normal sample, so it can be well distinguished from the normal sample.

In summary, it can be concluded that the TSMAE with an LSTM codec and an added memory module significantly improve anomaly detection in time-series anomaly detection tasks.

2) Ablation Experiment: See Table III. The TSMAE without $\mathcal{L}_{sparse}$ means that the sparsity coefficient η in the loss is set to 0. That is, the no-trend addressing vector uses fewer memory items, which allows the model to obtain anomaly-like samples through complex combinations. The TSMAE without memory means that the memory module is removed and the decoder is reconstructed directly using the latent

TABLE III
AUC RESULTS FOR ABLATION OF THE TSMAE

Algorithm	ECG5000	Wafer	Arrhythmia
TSMAE without $\mathcal{L}_{sparse}$	0.6845	0.9437	0.7510
TSMAE without Memory	0.6301	0.9650	0.7427
TSMAE without LSTM	0.9509	0.6548	0.7513
TSMAE	**0.9516**	**0.9912**	**0.7513**

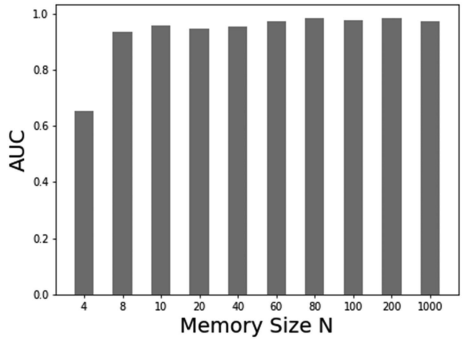


Fig. 5. TSMAE AUC at different memory module sizes.

representation generated by the encoder. The TSMAE without LSTM means that the LSTM layer of the codec is replaced with a fully connected layer. The results show that neither of the ablated models achieves the TSMAE, and the results of the ablated models are unstable in the three datasets. The TSMAE without memory achieves an AUC of 0.9650 on Wafer, which is also at the top of the excellent range compared to other algorithms but performs poorly on the ECG5000 dataset. The TSMAE without LSTM, in contrast, is good on ECG5000, differing only by 0.007 from the TSMAE, which has the highest AUC, but performs poorly on Wafer and is in the lowest position compared to all algorithms. The TSMAE maintains the highest AUC value in the three datasets. This is because the generalization suppression capability of the memory module presupposes that the model can reconstruct normal samples well; otherwise, normal and abnormal samples with equally large reconstruction errors are still close to each other and difficult to distinguish. Therefore, a combination of good codec LSTM is needed to produce a representative latent representation. While leaving the excellent codec alone is prone to overgeneralization, a memory module is needed to produce a suppression effect. The combination of the $\mathcal{L}_{sparse}$-constrained memory module and the LSTM yields a robust TSMAE.

3) Memory Size: The size N of the memory module essentially extracts N typical normal patterns, combining multiple normal patterns in different proportions to obtain the closest normal reconstruction sample to the original sample. Because of the sparsification of the addressing vector by Eqs. (17) and (21), the memory module does not have a long-tail distribution of addressing vector values at larger sizes. The model is encouraged to utilize as few memory items as possible to record

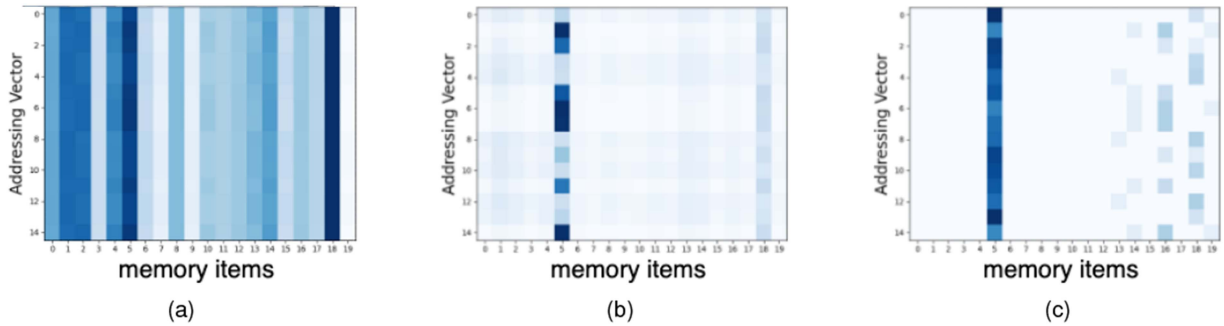


Fig. 6. Addressing vector with different sparsification operations. (a) Original (b) With $\mathcal{L}_{sparse}$ (c) With $\mathcal{L}_{sparse}$ and a rectification operation.

typical normal patterns. Therefore, the model is not sensitive to the size of the memory module. Fig. 5 shows the experimental results on the Wafer dataset by fixing the other parameters as described in Section 5.4 and only increasing the memory size. When the memory module size is satisfied with a sufficient number of typical features recorded, the memory module size increases significantly, while the AUC still tends to be flat.

4) Sparse Addressing Vector: The addressing vector becomes sparse under the induction of $\mathcal{L}_{sparse}$, and then the rectification of the vector less than the threshold λ to 0 by Equation (17) further constrains the addressing target to a certain number of strongly correlated memory items, excluding the rest of the miscellaneous terms with low similarity. The comparison in Fig. 6 shows the variation process of the two sparsification operations of $\mathcal{L}_{sparse}$ and the rectification operation to induce sparsity for the addressing vector on the ECG5000 dataset with a fixed memory size of 20, rectification threshold of $\frac{1}{N}$, and sparsification factor of 0.01. This experiment shows the addressing vector data distribution in the memory module. The horizontal coordinate represents the number of each memory item in the memory module with 20 memory items, and the vertical coordinate represents the number of the addressing vector corresponding to the data. Here, 14 represents the 14 test data randomly sampled. Fig. 6(a) shows the addressing vector without any sparsification operation. Fig. 6(b) shows the addressing vector with $\mathcal{L}_{sparse}$ induced sparsity, which shows that the addressing is more concentrated on individual memory terms, but there are still some smaller addressing coefficients. Fig. 6(c) shows the addressing vector with the rectification correction added, the smaller coefficients removed, and the addressing attention completely focused on individual memory items.

In loss $\mathcal{L}$, the sparsity of the addressing vector is controlled by adjusting the coefficients η of $\mathcal{L}_{sparse}$. The need for η varies for different datasets because different normal patterns have different meanings for the recorded features in the memory items. Fig. 7 shows the effect of trying η with values of $\{0,\ 1e^{-4},\ 1e^{-3},\ 1e^{-2},\ 1e^{-1},\ 1\}$ on the ECG5000 and Wafer datasets for the ROC according to the parameter settings in Chapter 5.4. ECG5000 prefers a larger induced sparse coefficient η, while Wafer prefers a medium induced coefficient η. This is due to the effect of the sample patterns in the two datasets. ECG5000 has a single normal sample pattern, with all normal samples having close values in the time series

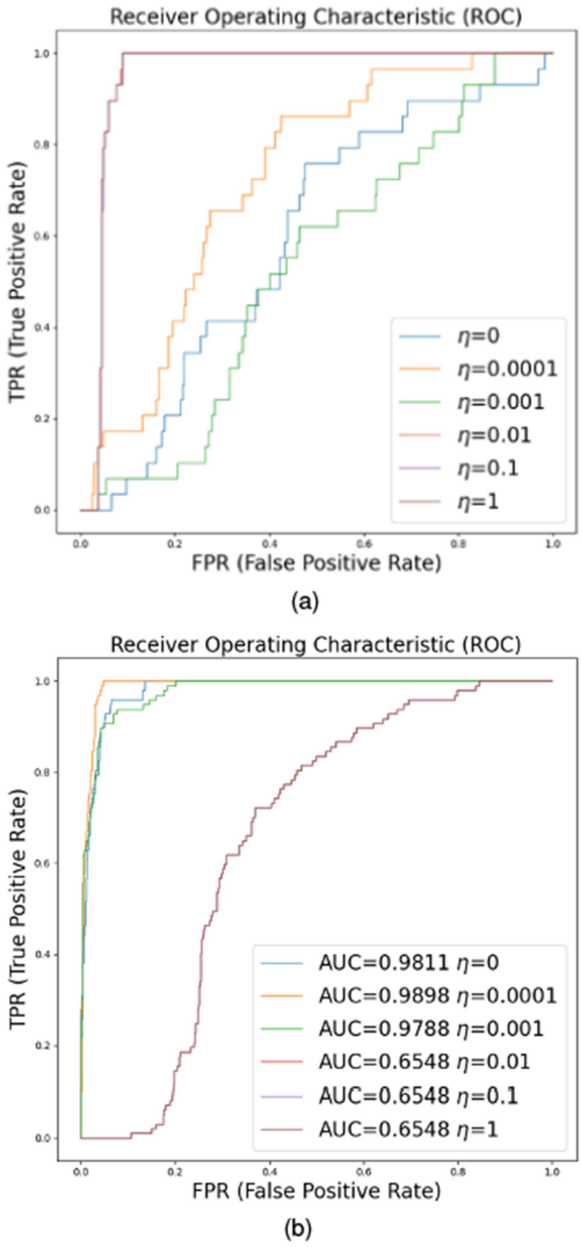


Fig. 7. ROC plots for different η, (a) for experimental results on ECG5000, (b) for experimental results on Wafer. (a) ECG5000 (b) Wafer.

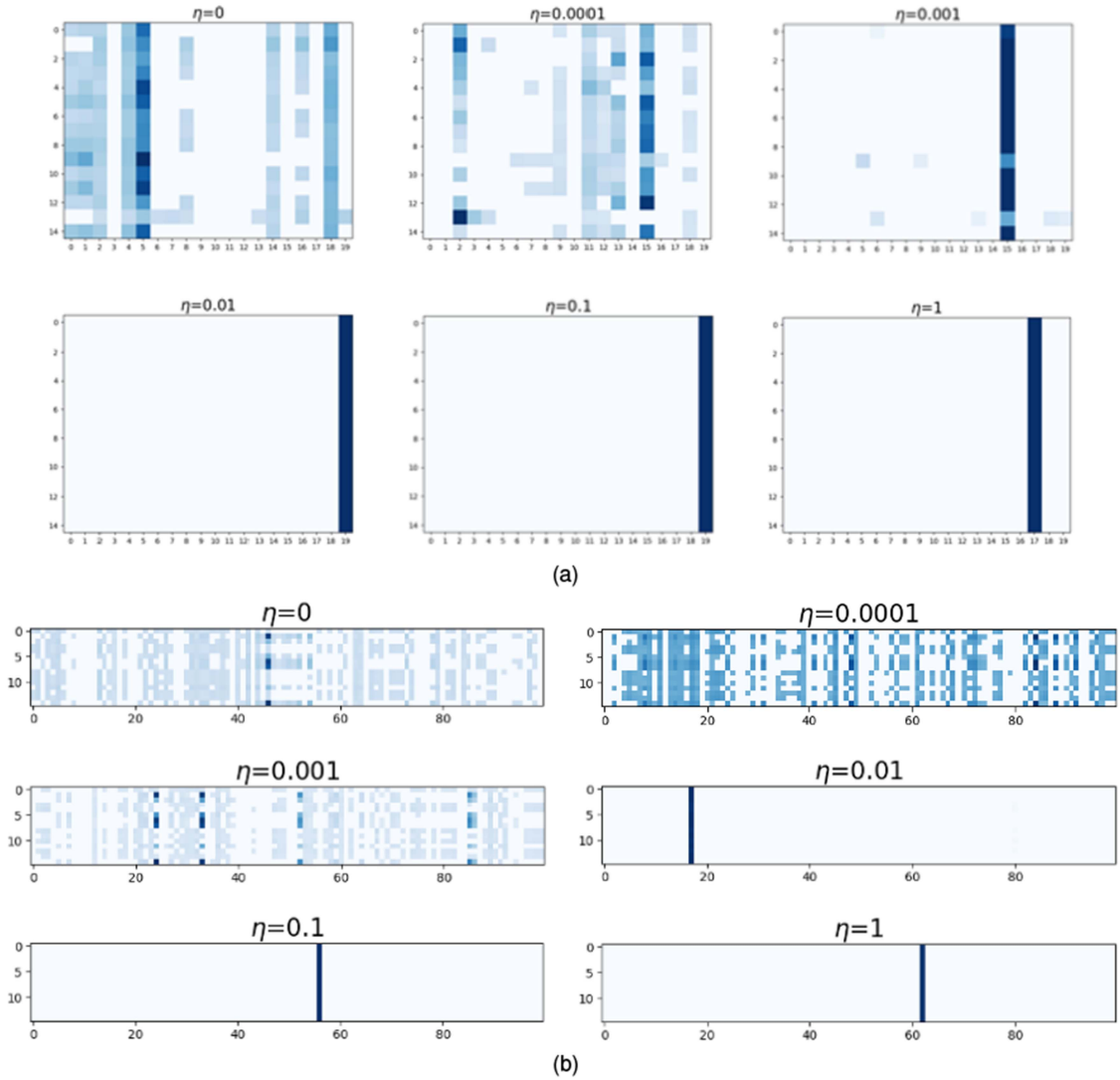


Fig. 8. Addressing vectors for different η, (a) experimental results on ECG5000, (b) experimental results on Wafer. X-axis indicates the number of memory items. (a) ECG5000 (b) Wafer.

and many abnormal patterns. Therefore, ECG5000 expects a higher induced sparsity coefficient to force the memory module to use fewer memory terms to learn a more homogeneous pattern. While the abnormal patterns in Wafer are clearly different from the normal patterns, there are multiple normal and abnormal patterns, so Wafer prefers not too many nor too few extreme memory terms to balance generalization ability with multiple memory patterns, allowing the model to suppress generalization while learning more of the latent representation patterns of normal samples.

Fig. 8 visualizes the addressing vector at different η, which better illustrates the different responses to the sparsification factor in the memory module for the ECG and Wafer datasets due to the different normal modes. For different data distributions and differences in the types of normal samples, and considering the existence of dataset quality, different datasets have different sparsity requirements for the memory module. The experiments in Fig. 8 are intended to build on the previous experiments and further demonstrate the differences in sparsity requirements by visualizing the results and giving a rough order-of-magnitude recommendation in a discussive way. It can be seen that in the ECG5000 dataset, due to the single shape of the normal mode, the prominence of addressing a single memory item is evident at the sparsification factor of 0.001, where only one memory item is used to record the entire normal mode, whereas it appears at 0.01 for Wafer. Therefore, for scenarios with a single normal mode, a larger sparsification factor above 0.001 should be chosen, and in scenarios with a large number of normal modes, a smaller sparsification factor below 0.001 can be chosen.

VI. Conclusion

Since the commonly used AE detection methods in IoT time-series anomaly detection tasks often overgeneralize, this paper designed a new TSMAE model. A memory module was added to the LSTM AE to suppress the generalization ability of the network so that the TSMAE can learn normal sample patterns and

eliminate abnormal sample patterns. In this way, the model does not generalize to well-reconstructed anomalies. The reconstruction errors of normal and anomalous samples are then different and can be easily distinguished by thresholding. This paper demonstrates the effectiveness and superiority of TSMAE in IoT time-series anomaly detection through experiments.

In future work, we can attempt to use sliding time windows, combined with a memory module as a dynamic knowledge base, to complete the anomaly detection task for streaming data and improve the real-time performance. Delaying the judgment by a few time windows enables the prediction task to be analogous to the reconstruction task, which can fully transpose the TSMAE design in this paper to the streaming anomaly detection scenario. Because multiple time windows will create more kinds of patterns, a memory module with multiple computational hops can be introduced. For temporal IoT data, the transformer will be used for temporal feature extraction. A pretraining method will be proposed to initialize the memory items of the transformer and memory module simultaneously.

REFERENCES

[1] M. James *et al.*, "The Internet of Things: Mapping the value beyond the hype," *McKinsey Glob. Inst.*, vol. 3, pp. 1–24, 2015.

[2] M. Hung, "Leading the IoT, Gartner insights on how to lead in a connected world," *Gartner Res.*, vol. 1, pp. 1–5, 2017.

[3] H. Wu, H. Tian, G. Nie, and P. Zhao, "Wireless powered mobile edge computing for industrial Internet of Things systems," *IEEE Access*, vol. 8, pp. 101 539–101 549, 2020.

[4] M. Pan *et al.*, "DHPA: Dynamic human preference analytics framework: A case study on taxi drivers' learning curve analysis," *ACM Trans. Intell. Syst. Technol.* vol. 11, no. 1, pp. 1–19, Jan. 2020. [Online]. Available: https://doi.org/10.1145/3360312

[5] D.-S. Pham, S. Venkatesh, M. Lazarescu, and S. Budhaditya, "Anomaly detection in large-scale data stream networks," *Data Mining Knowl. Discov.*, vol. 28, no. 1, pp. 145–189, 2014.

[6] M. Hasan, M. M. Islam, M. I. I. Zarif, and M. Hashem, "Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches," *Internet Things*, vol. 7, 2019, Art. no. 100059.

[7] H. Lu, C. Jin, X. Helu, C. Zhu, N. Guizani, and Z. Tian, "AutoD: Intelligent blockchain application unpacking based on JNI layer deception call," *IEEE Netw.*, vol. 35, no. 2, pp. 215–221, 2021.

[8] N. Hu, Z. Tian, H. Lu, X. Du, and M. Guizani, "A multiple-kernel clustering based intrusion detection scheme for 5G and IoT networks," *Int. J. Mach. Learn. Cybern.*, vol. 12, no. 11, pp. 3129–3144, 2021.

[9] M. Behniafar, A. Nowroozi, and H. R. Shahriari, "A survey of anomaly detection approaches in Internet of Things," *ISC Int. J. Inf. Secur.*, vol. 10, no. 2, pp. 79–92, 2018.

[10] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM Comput. Surv.*, vol. 54, no. 2, pp. 1–38, 2021.

[11] B. Sharma, L. Sharma, and C. Lal, "Anomaly detection techniques using deep learning in IoT: A survey," in *Proc. IEEE Int. Conf. Comput. Intell. Knowl. Economy*, 2019, pp. 146–149.

[12] A. A. Cook, G. Mirli, and Z. Fan, "Anomaly detection for IoT time-series data: A survey," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6481–6494, Jul. 2019.

[13] Z. Gu, W. Hu, C. Zhang, H. Lu, L. Yin, and L. Wang, "Gradient shielding: Towards understanding vulnerability of deep neural networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 2, pp. 921–932, Apr.–Jun. 2021.

[14] M. A. Hayes and M. A. Capretz, "Contextual anomaly detection in big sensor data," in *Proc. IEEE Int. Congr. Big Data*, 2014, pp. 64–71.

[15] H. Gao, Y. Xu, Y. Yin, W. Zhang, R. Li, and X. Wang, "Context-aware QoS prediction with neural collaborative filtering for Internet-of-Things services," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4532–4542, May 2020.

[16] Z. Cheng, C. Zou, and J. Dong, "Outlier detection using isolation forest and local outlier factor," in *Proc. Conf. Res. Adaptive Convergent Syst.*, 2019, pp. 161–168.

[17] D. Deng, "Research on anomaly detection method based on DBSCAN clustering algorithm," in *Proc. IEEE 5th Int. Conf. Inf. Sci. Comput. Technol. Transp.*, 2020, pp. 439–442.

[18] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. IEEE Int. Conf. Data Mining*, 2008, pp. 413–422.

[19] J. Chen, S. Sathe, C. Aggarwal, and D. Turaga, "Outlier detection with autoencoder ensembles," in *Proc. SIAM Int. Conf. Data Mining*, 2017, pp. 90–98.

[20] T. Kieu, B. Yang, and C. S. Jensen, "Outlier detection for multidimensional time series using deep neural networks," in *Proc. 19th IEEE Int. Conf. Mobile Data Manage.*, 2018, pp. 125–134.

[21] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," 2019, *arXiv:1901.03407*.

[22] G. Spigler, "Denoising autoencoders for overgeneralization in neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 4, pp. 998–1004, Apr. 2020.

[23] A. F. Kamara, E. Chen, Q. Liu, and Z. Pan, "Combining contextual neural networks for time series classification," *Neurocomputing*, vol. 384, pp. 57–66, 2020.

[24] M. Goldstein and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data," *PLoS One*, vol. 11, no. 4, 2016, Art. no. e0152173.

[25] Y. Zhang, Y. Chen, J. Wang, and Z. Pan, "Unsupervised deep anomaly detection for multi-sensor time-series signals," *IEEE Trans. Knowl. Data Eng.*, to be published, doi: 10.1109/TKDE.2021.3102110.

[26] X. Zhou, Y. Hu, W. Liang, J. Ma, and Q. Jin, "Variational LSTM enhanced anomaly detection for industrial Big Data," *IEEE Trans. Ind. Informat.*, vol. 17, no. 5, pp. 3469–3477, May 2021.

[27] D. Wu, Z. Jiang, X. Xie, X. Wei, W. Yu, and R. Li, "LSTM learning with bayesian and gaussian processing for anomaly detection in industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 8, pp. 5244–5253, Aug. 2020.

[28] H. Meng, Y. Zhang, Y. Li, and H. Zhao, "Spacecraft anomaly detection via transformer reconstruction error," in *Proc. Int. Conf. Aerosp. Syst. Sci. Eng.* 2019, pp. 351–362.

[29] C. Yin, S. Zhang, J. Wang, and N. N. Xiong, "Anomaly detection based on convolutional recurrent autoencoder for IoT time series," *IEEE Trans. Syst, Man, Cybern. Syst.*, vol. 52, no. 1, pp. 112–122, Jan. 2020.

[30] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "LSTM-based encoder-decoder for multi-sensor anomaly detection," 2016, *arXiv:1607.00148*.

[31] C. Zhang *et al.*, "A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 1409–1416.

[32] L. Li, J. Yan, H. Wang, and Y. Jin, "Anomaly detection of time series with smoothness-inducing sequential variational auto-encoder," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 3, pp. 1177–1191, Mar. 2021.

[33] J. Weston, S. Chopra, and A. Bordes, "Memory networks," in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1–15.

[34] S. Sukhbaatar, A. szlam, J. Weston, and R. Fergus, "End-to-end memory networks," *Proc. Int. Conf. Neural Inf. Process. Syst.*, vol. 2015, pp. 2440–2448, 2015.

[35] T. Han, W. Xie, and A. Zisserman, "Memory-augmented dense predictive coding for video representation learning," in *Proc. Comput. Vis. 16th Eur. Conf.*, 2020, pp. 312–329.

[36] B. Milde and C. Biemann, "SPARSESPEECH: Unsupervised acoustic unit discovery with memory-augmented sequence autoencoders," in *Proc. Int. Speech Commun. Assoc.*, 2019, pp. 256–260.

[37] Y.-Y. Chang, F.-Y. Sun, Y.-H. Wu, and S.-D. Lin, "A memory-network based solution for multivariate time-series forecasting," 2018, *arXiv:1809.02105*.

[38] Y. Zhu, W. Zhang, Y. Chen, and H. Gao, "A novel approach to workload prediction using attention-based LSTM encoder-decoder network in cloud environment," *EURASIP J. Wireless Commun. Netw.*, vol. 2019, no. 1, pp. 1–18, 2019.

[39] D. Gong *et al.*, "Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1705–1714.

[40] N. Hurley and S. Rickard, "Comparing measures of sparsity," *IEEE Trans. Inf. Theory*, vol. 55, no. 10, pp. 4723–4741, Oct. 2009.

[41] Y. Chen, Y. Hao, T. Rakthanmanon, J. Zakaria, B. Hu, and E. Keogh, "A general framework for never-ending learning from time series streams," *Data Mining Knowl. Discov.*, vol. 29, no. 6, pp. 1622–1664, 2015.

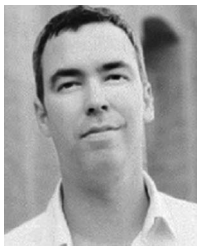
[42] R. T. Olszewski, *Generalized Feature Extraction for Structural Pattern Recognition in Time-Series Data*. Carnegie Mellon Univ., 2001.

[43] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[44] B. Schälkopf *et al.*, "Support vector method for novelty detection," in *Proc. Adv. neural Informat. Process. Syst.*, 1999, pp. 582–588.
[45] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2013, *arXiv:1312.6114*.
[46] L. Ruff *et al.*, "Deep one-class classification," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4393–4402.
[47] Y. Liu *et al.*, "Generative adversarial active learning for unsupervised outlier detection," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 8, pp. 1517–1528, Aug. 2020.
[48] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Statist. Workshop Conf. Proc.*, 2010, pp. 249–256.

Honghao Gao (Senior Member, IEEE) is currently with the School of Computer Engineering and Science, Shanghai University, Shanghai, China. He is also a Professor with Gachon University, Seongnam, South Korea. Prior to that, he was a Research Fellow with Software Engineering Information Technology Institute, Central Michigan University, Mount Pleasant, MI, USA, and was an Adjunct Professor with Hangzhou Dianzi University, Hangzhou, China. He has publications in IEEE TII, IEEE T-ITS, IEEE TNNLS, IEEE TSC, IEEE TFS, IEEE TNSE, IEEE TNSM, IEEE TCCN, IEEE TGCN, IEEE TCSS, IEEE TETCI, IEEE/ACM TCBB, IEEE IoT-J, IEEE JBHI, *IEEE Network*, ACM TOIT, ACM TOMM, ACM TOSN, and ACM TMIS. His research interests include software formal verification, industrial IoT networks, vehicle communication, and intelligent medical image processing. He was the recipient of the Best Paper Award at IEEE TII and EAI CollaborateCom 2020. Prof. Gao is a Fellow of Institution of Engineering and Technology (IET), a Fellow of British Computer Society (BCS), and is elected as a Member of the European Academy of Sciences and Arts (EASA). He is the Editor-in-Chief of the *International Journal of Intelligent Internet of Things Computing* (IJIITC), the Editor of *Wireless Network* (WINE) and *IET Wireless Sensor Systems* (IET WSS), and an Associate Editor for the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS (IEEE T-ITS), *IET Intelligent Transport Systems* (IET ITS), *IET Software*, *International Journal of Communication Systems* (IJCS), *Journal of Internet Technology* (JIT), and *Engineering Reports* (EngReports). Moreover, he has broad working experience in cooperative industry-university-research. He is a European Union Institutions-appointed external expert for reviewing and monitoring EU Project, is a Member of the EPSRC Peer Review Associate College for U.K. Research and Innovation in the U.K., and a Founding Member of IEEE Computer Society Smart Manufacturing Standards Committee.

Binyang Qiu is currently working toward the M.S. degree in software engineering with the School of Computer Engineering and Science, Shanghai University, Shanghai, China. His research interests include machine learning and IoT.



Ramón J. Durán Barroso received the Telecommunication Engineering and Ph.D. degrees from the University of Valladolid, Valladolid, Spain, in 2002 and 2008, respectively. He is currently an Associate Professor with the University of Valladolid, where he is the Deputy Director of the Faculty of Telecommunication Engineering. He is the author of more than 100 articles in international journals and conferences. His current research interests include the use of artificial intelligence techniques for the design, optimization, and operation of future heterogeneous networks, mobile edge computing, and network function virtualization. He is the Coordinator of the Spanish Research Thematic Network (Go2Edge: Engineering Future Secure Edge Computing Networks, Systems and Services) composed of 15 entities.

Walayat Hussain received the Ph.D. degree from the University of Technology Sydney, Sydney, NSW, Australia. He is currently a Lecturer (Assistant Professor) with Victoria University, Melbourne, VIC, Australia. Before joining Victoria University, he was a Lecturer and Research Fellow with the Faculty of Engineering and Information Technology, University of Technology Sydney, for six years. He was an Assistant Professor with BUITEMS University, Quetta, Pakistan, for many years. He has authored or coauthored in various top-ranked ERA-A*, JCR/SJR Q1 journals, such as *The Computer Journal*, *Information Systems Journal*, *Information Sciences*, IJIS, FGCS, IEEE ACCESS, *Computers and Industrial Engineering*, MONET, Journal of AIHC, IEEE TETCI, IEEE TSC, IEEE TGCN, IJCS, and WCMC. He is an Associate Editor for *IET Communications* and was a guest editor in various Q1 journals. He was the recipient of multiple national and international research awards and recognitions. He was the recipient of the Best Paper Award at 3PGCIC 2015, 2016 Poland, South Korea, Ministry of Higher Education Govt. of Oman and FEIT HDR Publication Award by the UTS Australia.

Yueshen Xu (Member, IEEE) received the Ph.D. degree from Zhejiang University, Hangzhou, China. He is currently an Associate Professor with the School of Computer Science and Technology, Xidian University, Xi'an, China. He was a co-trained Ph.D. candidate with the University of Illinois Chicago, Chicago, IL, USA. He has authored or coauthored more than 40 papers at a series of international conferences and in journals. His research interests include mobile computing, the Internet of Things, edge computing, and recommender systems. He is also a reviewer and PC member of many journals and conferences, such as IEEE ICDCS, *Mobile Networks & Applications*, *Neurocomputing*, and *Knowledge-based Systems*.

Xinheng Wang (Senior Member, IEEE) received the B.E. and M.Sc. degrees in electrical engineering from Xi'an Jiaotong University, Xi'an, China, in 1991 and 1994, respectively, and the Ph.D. degree in electronics and computer engineering from Brunel University, Uxbridge, U.K., in 2001. He is currently a Professor with the School of Advanced Technology and the Head of the Department of Mechatronics and Robotics, Xi'an Jiaotong-Liverpool University (XJTLU), Suzhou, China. Prior to joining XJTLU, he was a Professor with different universities in the U.K. He is an Investigator or Co-Investigator of about 30 research projects sponsored from EU, U.K. EPSRC, Innovate U.K., China NSFC, and industry. The research has led to the publications of more than 180 referred articles, 15 granted patents, and development of a few commercial products in condition monitoring, wireless mesh networks, and user-centric routing and navigation for group users. The smart trolley he has developed with an industry partner is the first airport Internet of Things (IoT) product in the world to provide smart services to air passengers. His current research interests include industrial IoT, edge AI for industrial machine vision, indoor positioning, acoustic localization, communications and sensing, and personalized navigation and services for group travelers. He is a General Chair/Technical Programme Chair of CollaborateCom more than last five years and a guest editor for a number of international journals.