Valve Stiction Detection Using Multitimescale Feature Consistent Constraint for Time-Series Data

Kexin Zhang, Yong Liu, Yong Gu, Jiadong Wang, and Xiaojun Ruan

***Abstract*—Using neural networks to build a reliable fault detection model is an attractive topic in industrial processes but remains challenging due to the lack of labeled data. We propose a feature learning approach for industrial time-series data based on self-supervised contrastive learning to tackle this challenge. The proposed approach consists of two components: data transformation and representation learning. The data transformation converts the raw time-series to temporal distance matrices capable of storing temporal and spatial information. The representation learning component uses a convolution-based encoder to encode the temporal distance matrices to embedding representations. The encoder is trained using a new constraint called multitimescale feature consistent constraint. Finally, a fault detection framework for the valve stiction detection task is developed based on the feature learning method. The proposed framework is evaluated not only on an industrial benchmark dataset but also on a hardware experimental system and real industrial environments.**

***Index Terms*—Hardware experimental system, industrial time-series, practical application, self-supervised learning, valve stiction detection.**

I. Introduction

VALVE stiction detection has always been an essential issue in the control loop performance assessment of process industries [1], [2]. Strong stiction causes unexpected oscillations that increase energy consumption and accelerate equipment wear. Reliable detection results rely on the features extracted from the industrial time-series data of control loops. Traditional methods for feature extraction are based on prior knowledge or the mechanism of a particular process. However, sometimes the full prior knowledge is not available in practice. In recent years, intelligent factories have received increased attention, and it is easier to collect industrial data than ever before [3]. Therefore, using neural networks to learn features from time-series data is a promising strategy to deal with various tasks in process industries. It avoids the time-consuming and labor-intensive process of manually extracting features using prior knowledge. It also provides a potential opportunity to achieve stiction detection using the neural-networks-based model.

According to the review [1], the traditional detection methods can be broadly classified into four categories, cross-correlation-function-based [4], limit-cycle-patterns-based [5], [6], nonlinearity-detection-based [7], and waveform-shape-based [8], [9]. This article reviews valve stiction detection methods from the perspective of the feature learning paradigm, i.e., manual feature engineering (MFE) and automatic learning representation (ALR). The MFE-based methods have advantages in interpretability, but they tend to be time-consuming, and reliable prior knowledge is required. Many MFE-based detection methods have been developed, such as the features derived from time-series data characterization [4], [10], [11], and the features derived from the shape of the controller output (OP) and process variable (PV) [6], [9], [12]. In contrast, ALR-based methods use learning algorithms on the collected data without the intervention of domain experts, directly extracting features [13], [14], [15]. The above ALR-based methods utilize neural networks' powerful feature representation capability and achieve higher accuracy than MFE-based methods.

The ALR-based stiction detection approaches require massive labeled data to train the networks. Our previous work used the MATLAB simulation to generate more labeled data for training the neural networks [15]. However, simulation is sometimes not feasible because complex industrial processes are challenging to model accurately. In practice, obtaining high-quality labeled data is time-consuming and expensive. Moreover, industrial processes are complex, and operators may not accurately understand the mechanism, leading to an obstacle to obtaining reliable labeled data. Moreover, the ALR-based approaches for control valve stiction detection are still being developed, to the best of our knowledge.

As one of the powerful techniques for feature learning, deep learning (DL) directly extracts features from the data and reduces the efforts in the design of hand-crafted features. Several

Manuscript received 21 April 2022; revised 26 August 2022 and 21 November 2022; accepted 4 December 2022. Date of publication 28 December 2022; date of current version 16 June 2023. Recommended by Technical Editor B. Chu and Senior Editor Y. Li. This work was supported by the National Key R&D Program of China under Grant 2021YFB2012300. *(Corresponding author: Yong Liu.)*

Kexin Zhang, Yong Liu, and Yong Gu are with the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310027, China (e-mail: zhangkexin@zju.edu.cn; yongliu@iipc.zju.edu.cn; gyxm@zju.edu.cn).

Jiadong Wang and Xiaojun Ruan are with the Zhejiang Supcon Technology Co. Ltd., Hangzhou 310053, China (e-mail: wangjiadong1@supcon.com; ruanxiaojun@supcon.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TMECH.2022.3227960.

Digital Object Identifier 10.1109/TMECH.2022.3227960

DL-based methods have already been developed for valve stiction detection [13], [14], [16], but these approaches require lots of labeled data to train networks, which is not easy and time-consuming in practical control loops. As a potential technique, self-supervised learning learns representations from unlabeled data and is widely used in audio processing and computer vision domains [17]. Inspired by experiences in unsupervised learning, we focus on developing an ALR-based approach for industrial time-series data in this article, which can be applied to the valve stiction detection task.

Specifically, we first propose a data transformation method for time-series, which converts the raw time-series data to temporal distance matrices based on the dynamic time warping (DTW) distance between different sliced windows of the raw time-series. Second, a convolution-based encoder encodes the temporal distance matrices to the embedding representations by a new constraint: multitimescale feature consistent constraint (MTFCC). This constraint encourages the encoder to learn the consistent representations on distance matrices derived from the same time-series. Then the representations are taken as the inputs of a traditional classifier. Finally, a valve stiction detection framework is developed based on the proposed methods. The main contributions are as follows.

1) A temporal distance matrix transformation method for time-series is developed. It converts the raw time-series to temporal distance matrices capable of storing temporal and spatial information and can be used without prior knowledge.
2) A new self-supervised representation learning method for time-series is proposed. It uses a convolution-based encoder to encode the distance matrices to embedding representations, and the encoder is trained by a new constraint called MTFCC.
3) A general fault detection framework is introduced. It consists of an unsupervised feature learning module and a detection module. A particular industrial case (valve stiction detection) is discussed and demonstrates the effectiveness of the proposed framework.

The rest of this article is organized as follows. Section II introduces the data transformation method, the feature learning model for industrial time-series data, and the practical implementation of valve stiction detection task. The experiments are described in both Sections III and IV. Finally, Section V concludes this article.

II. Methodology

A. Notation

A summary of the notation used in this article is given in Table I.

TABLE I
NOTATION

Notation	Meaning
$\mathbf{X}_i \in \mathbb{R}^{L\times D}$	Single time-series sample
$\mathcal{X}$	Dataset containing all samples.
$\mathcal{X}_{label}$	Subdataset containing labeled samples
$\mathcal{X}_{unlabel}$	Subdataset containing unlabeled samples
$\mathcal{Y}_{label}$	Labels of $\mathcal{X}_{label}$
$\hat{\mathcal{X}}$	All transformed temporal distance matrices
$\hat{\mathbf{X}}_i$	Transformed temporal distance matrices of ith time-series sample
$\mathbf{x}_i^h \in \mathbb{R}^{D\times S\times S}$	hth transformation for $\mathbf{X}_i$
$\mathcal{R}$	Representations of all time-series samples
$\mathbf{R}_i$	Representations of ith time-series sample
$\mathbf{r}_i^h \in \mathbb{R}^Z$	Embedding vector of $\mathbf{x}_i^h$
$\mathbf{B} \in \mathbb{R}^{S\times S}$	Temporal distance matrix
$DTW(W_j, W_k)$	DTW distance between (W_j, W_k)
$\mathbf{X}_{k'}^{(\ell)}$	k'th feature map of the ℓth convolutional layer
$\mathbf{W}_{k',k}^{(\ell)}$	Kernel of the convolutional layer
$\mathbf{C}_{k'}^{(\ell)}$	Bias of the convolutional layer
$\mathbf{X}_{flatten} \in \mathbb{R}^{KWQ}$	Flattened feature maps
$\mathbf{W}_{linear}$	Weight of the linear transformation layer
$\mathbf{C}_{linear}$	Bias of the linear transformation layer
$\mathrm{Sim}(\mathbf{v}_j, \mathbf{v}_k)$	Dot product similarity between $(\mathbf{v}_j, \mathbf{v}_k)$
$\mathcal{S}_{[g,g']}$	Similarity between gth and g'th group
$\mathcal{L}_g$	Loss for the gth group
$\mathcal{L}_{MTFCC}$	Final training loss
$\mathcal{T}$	Data transformation operation
$f_{trained}$	Trained convolutional-based encoder
Clf	Trained classifier
$\mathbf{x}_{new}$	New test sample
y	Predicted label of $\mathbf{x}_{new}$

B. Problem Definition

Consider $\mathbf{X}_i \in \mathbb{R}^{L\times D}, i = 1, 2, \ldots, N$ to be a single time-series sample of length L and dimension D, where N represents the total number of samples, and $y_i \in \mathbb{R}$ is the true label of the time-series sample $\mathbf{X}_i$. The set $\mathcal{X} = \mathcal{X}_{label} \cup \mathcal{X}_{unlabel}$ contains nonstiction samples and stiction samples. $\mathcal{X}_{label}$ denotes the subset of time-series data with true labels and $\mathcal{X}_{unlabel}$ denotes the subset without true labels. Here the number of samples in $\mathcal{X}_{label}$ is denoted as M, where $M < N$. The goal is to build an effective and practical detection framework based on DL. It should not only avoid manual feature engineering but should also require only a small number of labeled data to train the model.

C. Detection Framework

The proposed detection framework consists of two main modules: an unsupervised feature learning module to learn representations for time-series data, and a detection module for fault detection. The overall schematic is shown in Fig. 1.

1) Representation Learning Module: This module learns useful representations in an unsupervised manner using all available data, including labeled data and unlabeled data. To achieve this goal, the raw time-series data $\mathcal{X}$ are first transformed into temporal distance matrices under H predefined time scales. The set of the transformed temporal distance matrices is denoted as $\hat{\mathcal{X}} = \{\hat{\mathbf{X}}_1, \hat{\mathbf{X}}_2, \ldots, \hat{\mathbf{X}}_N\}$. Each time-series sample is transformed H times, so $\hat{\mathbf{X}}_i = \{\mathbf{x}_i^1, \mathbf{x}_i^2, \ldots, \mathbf{x}_i^H\}, i \in 1, 2, \ldots, N$. The hth transformation for $\mathbf{X}_i$ is $\mathbf{x}_i^h \in \mathbb{R}^{D\times S\times S}, h \in 1, 2, \ldots, H$, where S is the size of the temporal distance matrix. Then a convolution-based encoder f_θ is trained to encode $\hat{\mathcal{X}}$ to the embedding features

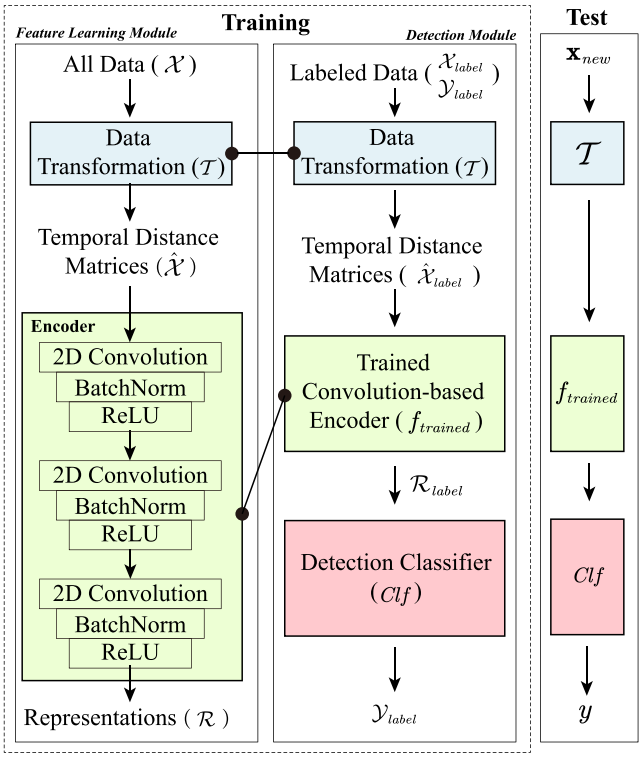


Fig. 1. Overall schematic of the proposed detection framework.

$\mathcal{R} = \{\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_N\}$ by using the proposed MTFCC. $\mathbf{R}_i$ denotes the set of representations of ith time-series sample at all time scales and $\mathbf{R}_i = \{\mathbf{r}_i^1, \mathbf{r}_i^2, \ldots, \mathbf{r}_i^H\}$. $\mathbf{r}_i^h \in \mathbb{R}^Z$ is the embedding vector for $\mathbf{x}_i^h$ and Z is the dimension of the embedding vector.

2) Detection Module: This module uses a small number of labeled data $\mathcal{X}_{label}$ to train a fault classifier Clf. The input to the classifier is the representations $\mathcal{R}$ that comes from the feature learning module, and the output of the classifier is the probabilities of possible outcomes $\mathcal{Y}$, such as nonstiction and stiction loop in stiction detection task.

D. Data Transformation for Time-Series

Data transformation is important since most data-driven approaches cannot directly deal with the raw data. For the stiction detection task, the proper transformation methods are crucial in recent DL-based approaches. For example, Kamaruddin et al. [13] transform the OP and PV data to the butterfly-shape-based images for valve stiction detection. Mohd Amiruddin et al. [14] proposed a method called D-value transformation and built a stiction detection network (SDN). The data transformation is also a useful component for other fault detection tasks, such as signal-to-image conversion method [18], multiple sensors stack method [19], and 2-D compressed construction method [20]. These approaches convert the raw time-series data to 2-D matrices or images capable of storing temporal and spatial features. Inspired by the above work, we develop a new data transformation method that transforms the time-series data into 2-D temporal distance matrices.

The proposed data transformation method constructs temporal distance matrices based on the distance measures between different time windows. The distance measure in time-series is an important topic. Recent work categorized time-series distance measure methods into two types, including lock-step measure and elastic measure. Lock-step measure refers to those distances that compare the ith point of one series to the ith point of another, while elastic measure aims to create a nonlinear mapping in order to align the series and allow comparison of one-to-many points [21]. The typical lock-step measure is widely used in many domains, called Euclidean distance. To address the issue of time-step alignment, elastic measure approaches were developed [22], such as DTW distance, derivative DTW distance, weighted DTW distance, edit distance (ED), and time warp edit distance (TWED). Based on the experimental results in [22], the DTW distance is used as the measurement method because it is relatively simple, and no significant performance difference is observed in the above elastic measures. Moreover, Euclidean distance is also a particular case of DTW distance.

1) Temporal Distance Matrix Transformation on Univariate Time-Series: The transformation method for a univariate time-series is first introduced, and then it is generalized to multivariate time-series. Consider $\mathbf{X}_i \in \mathbb{R}^{L\times D}$ to be a univariate time-series sample when $D = 1$. For simplicity, the subscript i is removed in the following description. Our goal is to transform this series into a temporal distance matrix $\mathbf{B}$, where $\mathbf{B} \in \mathbb{R}^{S\times S}$. S is the size of temporal distance matrix.

First, the time-series $\mathbf{X}$ is sliced into S nonoverlapping windows, and the length of each window is allowed to be different. The set of the consecutive windows is defined as $\mathcal{W}$, where... Then, the temporal distance matrix is defined as

$$\mathbf{B}(j,k) = DTW(W_j, W_k) \cdot \frac{1}{|j-k|} \tag{1}$$

where the right side of (1) is the temporal distance between jth window and kth window, and a total of S^2 distance values can be obtained. $DTW(\cdot)$ denotes DTW distance measure. Although the first term can measure the similarity between any two windows, their temporal information is not considered. Consider that the distance of two consecutive windows is more likely to close, the second term $\frac{1}{|j-k|}$ provides a temporal weighing mechanism between jth window and kth window. This term is also consistent with the assumption widely used in time-series data [23], [24], [25], i.e., given a window as an anchor, the windows adjacent to it are more likely to be similar, and the windows further away should be less similar. Similarity indicates that two windows have more common patterns, such as the same amplitude, the same periodicity, or the same trend.

Because the window length is randomly determined, there may be an extreme case, i.e., the length of a window is $L - S + 1$, while the length of the rest $S - 1$ windows is 1. Therefore, a new window slicing method is employed to avoid the extreme cases. Note that S windows require $S + 1$ sliced timestamps.

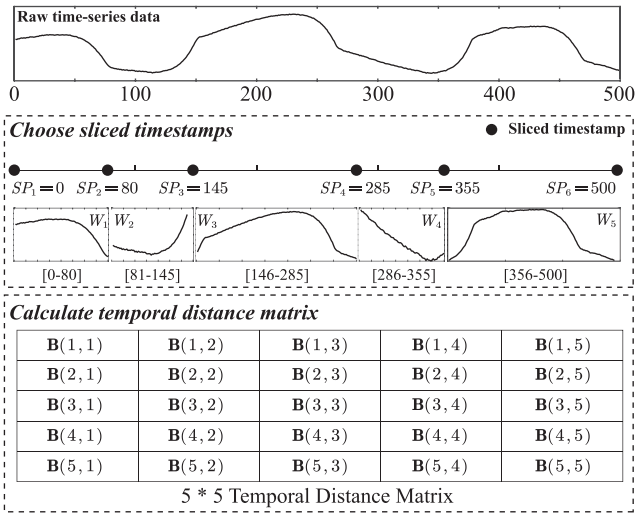


Fig. 2. Temporal distance matrix transformation on univariate time-series.

The first sliced timestamp is set to 0 and the last sliced timestamp is set to L in this article. The rest $S-1$ sliced timestamps are calculated by

$$SP_s = (s-2) \times L_{avg} + random(1, L_{avg}),\ s = 2, 3, \ldots, S \tag{2}$$

where $random(1, L_{avg})$ is a random sampling operation that randomly sample a positive integer between [1,L_{avg}]. L_{avg} is defined as

$$L_{avg} = \left\lfloor \frac{L}{S} \right\rfloor \tag{3}$$

where $\lfloor \cdot \rfloor$ is an operation that returns the largest integer not greater than $\frac{L}{S}$.

An example of temporal distance matrix transformation for univariate time-series is shown in Fig. 2. A univariate time-series of length 500 is considered, and the size of the temporal distance matrix is set to 5. First, the raw time-series is sliced into five windows using points 0, 80, 145, 285, 355, and 500. Then, the temporal distance between any two windows is calculated by (1). Finally, the temporal distance matrix is formed by filling each distance value to the corresponding position.

2) Extend to Multivariate Time-Series: The transformation method mentioned in Section II-D1 is designed for univariate time-series. However, multivariate time-series need to be processed in most industrial tasks. Therefore, a simple extended version of the transformation method is provided. Specifically, (1) is redefined as

$$\mathbf{B}(d, j, k) = DTW\left(W_j^d, W_k^d\right) \cdot \frac{1}{|j-k|},\ d = 1, 2, \ldots, D. \tag{4}$$

Specifically, the transformation method for multivariate time series is divided into two steps. The first step is calculating the temporal distance matrix in each dimension, and then all matrices are stacked in the second step.

E. Feature Learning With MTFCC

Based on the proposed detection framework, an encoder is required to encode temporal distance matrices to informative representations. As one of the successful DL models, convolution-based neural networks (CNNs) models have made breakthroughs in 2-D image recognition. At the same time, the dimension of temporal distance matrices is also 2-D, so the convolution-based encoder is preferred in this article. Moreover, CNNs are widely used in industrial fault detection tasks, such as identification of complex power quality disturbances [26], bearing fault diagnosis [18], rotating machinery diagnosis [19], and gearbox fault diagnosis [27]. These successful applications show the effectiveness of the DL in industrial fault detection, especially the CNN-based approaches. Therefore, a convolution-based encoder is adopted to encode the 2-D temporal distance matrices to embedding representations in this article.

1) Convolution-Based Encoder: The encoder is formed by stacking multiple convolutional layers. In a single 2-D convolutional layer, the input is convolved with a set of learnable kernels to produce outputs (also known as feature maps) as the input to the next layer [19]. The operation of ℓ-th convolutional layer can be expressed as

$$\mathbf{X}_{k'}^{(\ell)} = \sum_{k=1}^{K} \mathbf{W}_{k',k}^{(\ell)} * \mathbf{X}_k^{(\ell-1)} + \mathbf{C}_{k'}^{(\ell)},\ k' = 1, 2, \ldots, K' \tag{5}$$

where $\mathbf{X}_{k'}^{(\ell)}$ is the k'th feature map in the output of the ℓth convolutional layer, and $\mathbf{X}_k^{(\ell-1)}$ is the kth feature map in the output of the $(\ell-1)$th convolutional layer. K' and K are the number of feature maps in the outputs of the ℓth layer and the $(\ell-1)$th layer, respectively. $*$ is the convolution operator. $\mathbf{W}_{k,'k}^{(\ell)}$ is the k'th convolutional kernel that applied to $\mathbf{X}_k^{(\ell-1)}$. $\mathbf{C}_{k'}^{(\ell)}$ is a bias. Generally, a nonlinear activation function is applied to feature maps for increasing the nonlinear property. In this article, rectified linear unit (ReLU) function is used since it is widely used and proved to work far better in most of the classification tasks for its capacity to accelerate the convergence and alleviate the vanishing gradient problem [28]. ReLU is an element-wise operation and it is defined as

$$ReLU(\mathbf{X}_k(i,j)) = max(0, \mathbf{X}_k(i,j)) \tag{6}$$

where $\mathbf{X}_k(i,j)$ is the element at ith row and jth column of kth feature map.

Suppose the output of the last convolutional layer is $\mathbf{X}_{out}$, which contains K 2-D feature maps, and $\mathbf{X}_k \in \mathbb{R}^{W \times Q}$ is the kth feature map. The size a feature map is $W \times Q$. Then, the flatten operation is applied to these feature maps for obtaining the 1-D embedding vector $\mathbf{X}_{flatten} \in \mathbb{R}^{KWQ}$. The flatten operation is defined as

$$\mathbf{X}_{flatten}(k * HW + i * W + j) = \mathbf{X}_{out}(k, i, j) \tag{7}$$

which means that the element at ith row and jth column of kth feature map is assigned to the $(k * HW + i * W + j)$th element of $X_{flatten}$. A linear transformation layer is further used to map the embedding vectors into Z-dimensional vectors. The linear

transformation is expressed as

$$\mathbf{r} = \mathbf{W}_{linear}\mathbf{X}_{flatten} + \mathbf{C}_{linear} \tag{8}$$

where $\mathbf{r}$ is the final representation of a given time-series sample, and $\mathbf{X}_{flatten}$ is the flattened feature maps. $\mathbf{W}_{linear} \in \mathbb{R}^{Z_{flatten} \times Z}$ is the learnable weights of the linear transformation layer, where $Z_{flatten}$ is the dimension of the flattened feature maps. $\mathbf{C}_{linear} \in \mathbb{R}^{Z}$ is a bias that makes representations more informative. The encoder without training is defined as f_θ in this article.

2) Multitimescale Feature Consistent Constraint (MTFCC): Suppose we have two representations $\mathbf{r}_i^h$ and $\mathbf{r}_i^{h'}$, they come from the same time-series sample but with time scale h and h', respectively. The consistent constraint expects that the representations of the temporal distance matrices derived from one time-series sample are consistent even if they are generated under the different times scales. In contrast, the representations of the temporal distance matrices derived from different time-series are inconsistent. In other words, the goal is to maximize the similarity between $\mathbf{r}_i^h$ and $\mathbf{r}_i^{h'}$, and minimize the similarity between $\mathbf{r}_i^h$ and $\mathbf{r}_{i'}^h$, where $h, h' = 1, 2, \ldots, H$ and $i, i' = 1, 2, \ldots, N$. In this article, the similarity is defined as the dot product between the L_2 normalized $\mathbf{r}_i^h$ and $\mathbf{r}_i^{h'}$ (or $\mathbf{r}_i^h$ and $\mathbf{r}_{i'}^h$), i.e.,

$$\mathrm{Sim}(\mathbf{r}_i^h, \mathbf{r}_i^{h'}) = \mathbf{r}_i^{h\top}\mathbf{r}_i^{h'} / \|\mathbf{r}_i^h\| \|\mathbf{r}_i^{h'}\|. \tag{9}$$

At the training stage, we first randomly sample B temporal distance matrices and split them into different groups based on the following rule: the matrices in each group should come from the same original time-series sample. Suppose there are G groups and gth group is considered as an anchor, the distance matrices in the gth group should be similar, and the distance matrices in the rest $G-1$ groups should not be similar to those in the gth group. Specifically, the similarity of samples in the gth group is defined as the positive similarity. Suppose g'th group is any one of the rest $G-1$ groups. The similarity between gth group and the g'th group is defined as the negative similarity. Two similarities are denoted as

$$\mathcal{S}_{[g,g']} = \begin{cases} \frac{2}{N_g}\sum_{j=1}^{\lfloor N_g/2 \rfloor} \mathrm{Sim}(\mathbf{r}_g^j, \mathbf{r}_g^{N_g-j}), & g' = g \\ \mathrm{Sim}(\tilde{\mathbf{r}}_g, \tilde{\mathbf{r}}_{g'}), & g' \neq g \end{cases} \tag{10}$$

where N_g is the number of samples in the gth group. $\tilde{\mathbf{r}}_g$ and $\tilde{\mathbf{r}}_{g'}$ denotes the average representation of all distance matrices in gth group and g'th group, respectively. They are defined as

$$\tilde{\mathbf{r}}_g = \frac{1}{N_g}\sum_{i=1}^{N_g}\mathbf{r}_g^i, \quad \tilde{\mathbf{r}}_{g'} = \frac{1}{N_{g'}}\sum_{i=1}^{N_{g'}}\mathbf{r}_{g'}^i. \tag{11}$$

The loss for the gth group is defined as

$$\mathcal{L}_g = -\log \frac{\exp(\mathcal{S}_{[g,g]})}{\sum_{g'=1}^{G}\exp(\mathcal{S}_{[g,g']})}. \tag{12}$$

The final training loss is the average of the losses for each group, which is defined as

$$\mathcal{L}_{MTFCC} = \frac{1}{G}\sum_{g}^{G}\mathcal{L}_g. \tag{13}$$

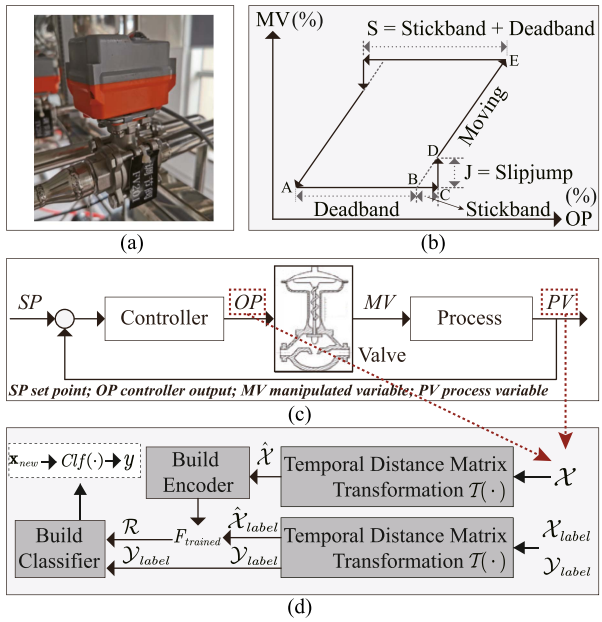


Fig. 3. Typical behavior of valve stiction in a control loop and practical implementation of stiction detection. (a) Real valve. (b) Typical behavior of valve stiction. (c) Closed-loop control system. (d) Practical implementation of stiction detection.

$\mathcal{L}_{MTFCC}$ is the loss in the training stage. Since it is designed based on the principle of consistent constraint under multiple time scales, $\mathcal{L}_{MTFCC}$ is also called MTFCC in this article. The whole training stage works in an unsupervised manner, and the goal is to obtain the best encoder f_θ that maps the temporal distance matrices to the informative representations. The trained encoder is defined as $f_{trained}$.

F. Practical Implementation

In this section, the practical implementation of valve stiction detection is introduced. The overall schematic is shown in Fig. 3.

1) Valve Stiction Behavior: A valve is an actuator in a control system and its stiction behavior has been described in [29]. Specifically, the typical stiction behavior is shown using the phase plot of valve output [manipulated variable (MV)] and OP. As shown in Fig. 3(b), when a valve comes to rest or changes the direction at point A, the valve sticks. As the OP increases, MV does not change because of the deadband AB and the stickband BC. When OP overcomes the deadband and the stickband, the value of MV suddenly jumps to the new position D because of the potential energy stored in the actuator and starts to move. *S* and *J* quantify the stiction behavior in the two-parameters data-driven valve stiction model, where $S = deadband + stickband$ and $J = slipjump$. In this data-driven model, there is a linear look-up table that translates the control signal (mA) to the percentage of valve travel in this data-driven model. Therefore, *S* is the translated percentage, measured as a %. In this data-driven model, *J* is also measured as a %, accounting for the offset between the valve input and output signals. For example, if $S = J$, there is no offset between the input and output. Once the valve overcomes stiction, valve output tracks the valve input exactly. The final output of

this data-driven model is again converted back to a mA signal using a lookup table based on the valve characteristics [5], [29]. In practice, MV is not available in some traditional valves, so PV is selected as an alternative [9], [13], [14].

2) Stiction Detection: A classic closed-loop control system is shown in Fig. 3(c) and the practical implementation is shown in Fig. 3(d). In practice, each sample is a 2-D time-series data containing two variables: OP and PV. The raw OP/PV data are in sequence with time stamps denoted by $t_1, t_2, \ldots, t_L$, and OP and PV can be denoted by $\mathbf{x}_{op} = \left\{x_{op}^{t_1}, x_{op}^{t_2}, \ldots, x_{op}^{t_L}\right\}$ and $\mathbf{x}_{pv} = \left\{x_{pv}^{t_1}, x_{pv}^{t_2}, \ldots, x_{pv}^{t_L}\right\}$, respectively.

The first step is to obtain the temporal distance matrices for all available samples, here, the transformation operation mentioned in Section II-D is defined as $\mathcal{T}(\cdot)$, so

$$\hat{\mathcal{X}} = \mathcal{T}(\mathcal{X}). \tag{14}$$

Then taking $\hat{\mathcal{X}}$ as the inputs, a convolution-based encoder can be trained using the proposed constraint MTFCC. The trained encoder is taken as the feature extractor and is defined as $f_{trained}$.

Next the classifier for stiction detection is built using $\mathcal{X}_{label}$ and the corresponding labels $\mathcal{Y}_{label}$. The inputs of the classifier is defined as

$$\mathcal{R} = f_{trained}(\mathcal{T}(\mathcal{X}_{label})). \tag{15}$$

Using $\mathcal{R}$ as the inputs and $\mathcal{Y}_{label}$ as the outputs, the classifier can be built as

$$\mathcal{Y}_{label} = Clf(\mathcal{R}) \tag{16}$$

where Clf is the final classifier to achieve stiction detection. In our framework, the choice of classifier is flexible, such as neural networks, support vector machines, and other machine learning algorithms.

In practice, whether a new sample $\mathbf{x}_{new}$ is stiction can be determined by the following process:

$$y = Clf(f_{trained}(\mathcal{T}(\mathbf{x}_{new}))) \tag{17}$$

where y is the predicted label of the new sample $\mathbf{x}_{new}$. In the stiction detection task, it can be either 0 or 1, where 0 means nonstiction class and 1 means stiction class.

III. Experiments on Benchmark Dataset

A. International Stiction Data Base (ISDB)

The international stiction data base (ISDB) is a well-known benchmark for the validation of new techniques concerning valve stiction detection. These loops were collected from various process industries, including chemical plants (CHEM), pulp and paper mills (PAP), buildings (BAS), mining (MIN), and power plants (POW). Most stiction detection methods were evaluated on this public industrial dataset.

B. Parameter Settings

The parameters for the convolution-based encoder are as follows. The kernel size for all convolutional layers is set to 3. The number of convolutional layers is set to 3. Each layer of the encoder is a combination of a 2-D convolution layer, a batch normalization layer, and an ReLU nonlinear function. $Z_{flatten}$ is the dimension of the flattened feature map of the last convolutional layer The size of a temporal distance matrix is set to 28. 48 and 64 are also considered in the experiment, and we found that the detection framework has shown satisfactory performance when the matrix size is 28. Choosing 48 or 64 will increase the model's parameters and cause additional computational costs. Support vector machine (SVM) is used as the final classifier because SVM and its extensions are one class of the most successful machine learning methods in the past decades.

TABLE II
BENCHMARK INDUSTRIAL LOOPS

Loop	Type	Malfunction	Loop	Type	Malfunction
CHEM 1	Fic	Stiction	CHEM 24	Fic	Likely stiction
CHEM 2	Fic	Stiction	CHEM 26	Lev	Likely stiction
CHEM 3	Tem	Quantisation	CHEM 29	Fic	Stiction
CHEM 4	Lev	Tuning problem	CHEM 32	Fic	Likely stiction
CHEM 5	Fic	Stiction	CHEM 33	Fic	Disturbance
CHEM 6	Fic	Stiction	CHEM 34	Fic	disturbance
CHEM 10	Pre	Stiction	CHEM 58	Fic	No oscillation
CHEM 11	Fic	Stiction	MIN 1	Tem	Stiction
CHEM 12	Fic	Stiction	PAP 2	Fic	Stiction
CHEM 13	Ana	Faulty sensor	PAP 4	Con	Tight tuning
CHEM 14	Fic	Faulty sensor	PAP 5	Con	Stiction
CHEM 16	Pre	Interaction	PAP 7	Fic	Disturbance
CHEM 23	Fic	Likely stiction	PAP 9	Tem	No-stiction

TABLE III
COMPARISON OF DIFFERENT STICTION DETECTION METHODS

Method	Accuracy	NOT tested loops
Limit cycle patterns analysis [30]	2/2	24
MV(OP) patterns analysis [6]	12/14	12
MV(OP) patterns analysis [31]	14/16	10
Fuzzy clustering method [32]	2/6	20
Higher-order statistics method [7]	19/24	2
Cross-correlation method [4]	14/24	2
Statistics method [2]	16/25	1
Relay-based method [8]	17/26	0
Curve fitting method [9]	12/25	1
Waveform shape analysis [12]	11/26	0
PSD/ACF method [33]	18/26	0
Peak slope method [34]	14/25	1
Zone Segmentation Method [34]	15/25	1
BSD-CNN method [13]	20/26	0
D-value ANN method [14]	19/24	2
MTFCC-based method (Ours)	**22/26**	**0**

PSD/ACF: Power spectral density / Auto correlation function;
BSD CNN: Butterfly shape-based convolutional neural networks;
D-value ANN: D-value artificial neural network;

C. Results of Stiction Detection

The comparison results with other 15 stiction detection methods are listed in Table III. A total of 26 control loops were tested. The primary information of the test loops is described in Table II, where Tem, Fic, Pre, Lev, Con, Ana denote the temperature,

flow, pressure, level, concentration, and analyzer control loop, respectively.

In Table III, it can be seen that only six methods apply to all test loops. Most of the other established methods apply to a narrower range of processes. It first shows that our framework applies to a broader range of process loops. Moreover, the methods proposed in [13] and [14], also used DL techniques to build detection models. However, the detection method in [14] does not apply to CHEM 4 and CHEM 5, and the accuracy is lower than our method. Although the method proposed in [13] can be applied to all test loops, the accuracy is also lower than our method. Our method outperforms the recent DL-Based stiction detection with a total detection accuracy of 22/26, which is the highest one of the considered methods.

As shown in Table II, in 26 test control loops, 15 loops have been identified to have some form of stiction (including likely stiction loops), in which 13 loops are correctly identified in our method, except CHEM 5 and CHEM 29. Eleven out of 26 control loops are marked as no-stiction, and these loops are known to have other types of nonlinearities rather than stiction. Our method correctly identified nine out of 11 loops, except PAP 9 and CHEM 34. It can be concluded that the proposed method has 86.7% (13/15) accuracy on stiction loops and 81.8% (9/11) accuracy on no-stiction loops, and the overall detection accuracy is 84.6% (22/26).

D. Discussion

This subsection uses more data to comprehensively evaluate the proposed detection framework. First, 81 loops are selected in the ISDB dataset, of which 30 loops are randomly selected as the labeled loops, and the remaining 51 loops are unlabeled test loops. Then, three base classifiers are considered, SVM, Logistic regression (LR), and K-means clustering. Note that K-means is an unsupervised algorithm, but SVM and LR are supervised algorithms, so the different performance metrics are adopted. When using K-means as the base classifier, the clustering accuracy is defined as

$$Acc_u = \max_{perm\in\mathcal{P}} \frac{1}{n}\sum_{i=0}^{n-1} \mathbb{1}_{[perm(\hat{y}_i)=y_i]} \tag{18}$$

where P is the set of all permutations in $[1, 2, \ldots, K]$ where K is the number of clusters. Alternatively, the classification accuracy is used to evaluate the supervised algorithms, i.e.,

$$Acc_s = \frac{1}{n}\sum_{i=0}^{n-1} \mathbb{1}_{[\hat{y}_i=y_i]} \tag{19}$$

where $\mathbb{1} \in \{0, 1\}$ is the indicator function evaluating to 1 if $\hat{y}_i = y_i$. $\hat{y}_i$ is predicted label and y_i is the true label of ith time-series sample.

1) Effectiveness of MTFCC and Data Transformation: To show the effectiveness of MTFCC and the series-time transformation methods, the following experimental strategies are considered: 1) classifier + transformation + MTFCC; 2) classifier + transformation; and 3) classifier + raw data. The classifier can be set to one of SVM, LR, and K-means. We consider three different numbers of labeled data for each strategy: 30, 20, and 10. Therefore, the experimental settings include the number of labeled data, the classifier, whether to use data transformation, and whether to use MTFCC. For each strategy, we conduct 50 experiments to obtain the intervals of detection accuracy. For example, consider the following experimental setting: 30 labeled loops, SVM classifier, use data transformation, and use MTFCC. A total of 50 experiments were conducted. For each of the 50 experiments, the 30 labeled data are randomly selected as the training data, and the remaining data are test data. The detection results fall in interval [0.84, 0.98]. The results of three experimental strategies are shown in Fig. 4. It can be seen that the classifier combined with transformation and MTFCC achieves the highest accuracy in the three strategies, the classifier only combined with the transformation method gives moderate accuracy, and the classifier just combined with raw data gives the lowest accuracy. The results show that the transformed data obtained by the proposed time-series transformation method are more suitable for the classifier than the raw data, while the MTFCC-based framework can further encode the data to more informative representations. Although this article uses SMV, LR, and K-means as the final classifiers, other classifiers are also available in practice since the proposed MTFCC and time-series transformation methods constitute a feature extractor.

TABLE IV
STATISTICS UNDER THREE STRATEGIES

Model	Number of training loops					
	10		20		30	
	mean	std	mean	std	mean	std
MTFCC+Transformed+SVM	0.898	0.048	0.921	0.035	0.933	0.037
Transformed+SVM	0.787	0.069	0.860	0.054	0.870	0.054
Raw data+SVM	0.665	0.083	0.705	0.055	0.735	0.061
MTFCC+Transformed+LR	0.834	0.061	0.878	0.050	0.889	0.039
Transformed+LR	0.744	0.094	0.830	0.064	0.856	0.053
Raw data+LR	0.607	0.078	0.583	0.089	0.538	0.078
MTFCC+Transformed+K-means	0.737	0.082	0.774	0.058	0.789	0.049
Transformed+K-means	0.730	0.043	0.740	0.026	0.744	0.016
Raw data+K-means	0.672	0.063	0.669	0.046	0.649	0.046

2) Stabilization for Limited Training Loops: As mentioned before, obtaining sufficient and high-quality data is time-consuming and expensive in practice. Therefore, the stabilization of the proposed MTFCC and time-series transformation methods for limited training loops is also important. Therefore, the experiments using 10, 20, and 30 training loops are implemented. The experimental results and statistics under different training loops are shown in Table IV. It can be seen that the proposed methods are useful for different numbers of training data. For SVM and LR, the proposed MTFCC and transformation methods provided the highest accuracy and the lowest standard deviation.

3) Parameter Study: The proposed framework has three important parameters, including the number of time scales, the size of the temporal distance matrix, and the size of the convolutional kernel. Some experimental results are provided to show the effect

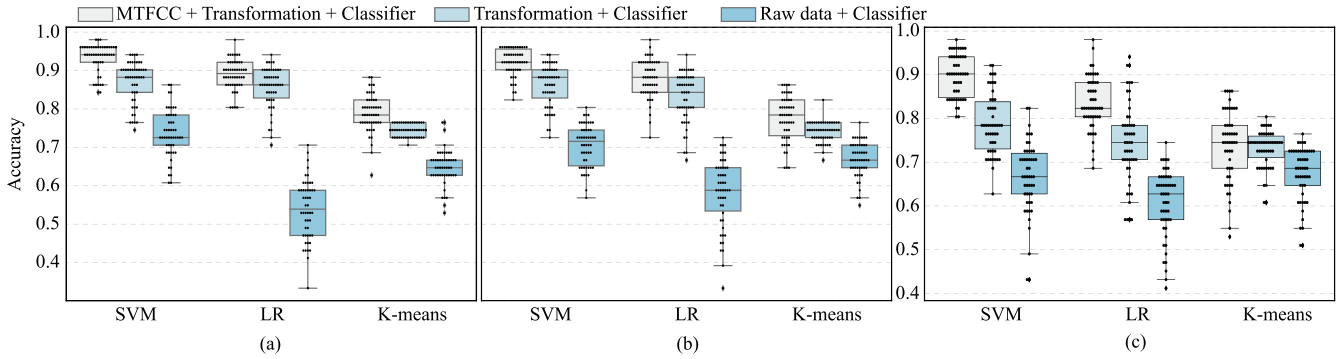


Fig. 4. Detection accuracy under three experimental strategies. Number of training loops. (a) 30. (b) 20. (c) 10.

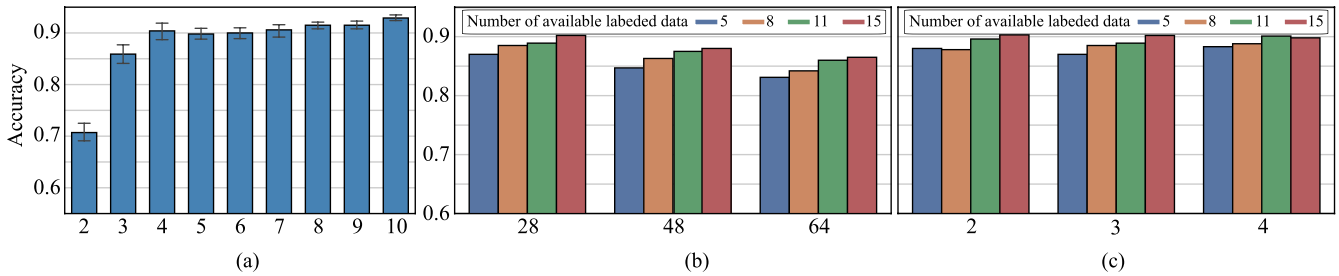


Fig. 5. Parameter study. (a) Different number of time scales are discussed in MTFCC. (b) Detection accuracy under different size of temporal distance matrix (28, 48, and 64). (c) Influence of different convolutional kernels on the detection accuracy.

of these parameters on the detection results, which are shown in Fig. 5.

The number of time scales is important for learning representations using MTFCC, the detection accuracy under the different number of time scales is summarized as Fig. 5(a). The results show that MTFCC benefits from the number of time scales, which means selecting more time scales make obtaining higher accuracy easier. We argue that more time scales provide a stronger contrastive constraint, which allows the model to learn more representations. However, as the number increases, the accuracy maintains a stable range.

The size of the temporal distance matrix is a parameter for the data transformation method. We considered three sizes, including 28, 48, and 64. The detection results are shown in 5(b). It can be seen that the detection accuracy decreases as the size of the temporal distance matrix increases. We argue that a larger distance matrix contains more useless information, such as noise. The nuisance information leads to the performance deterioration of the detection framework. Therefore, the size 28 is adopted in the final detection framework.

The kernel size is a parameter for the convolution-based encoder. Three values are considered, including 2, 3, and 4. On the one hand, it is difficult for a small convolutional kernel to extract global information unless increasing the depth of the model, which results in more computational cost. A large convolutional kernel is an alternative but it requires a larger input matrix. On the other hand, the detection results in Fig. 5(c) show that the size of the convolutional kernel has little effect on the detection accuracy. Therefore, the convolutional size is set to 3 in the detection framework.

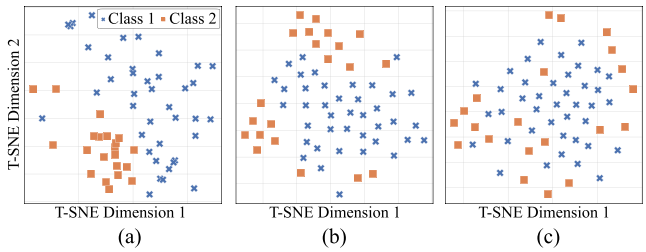


Fig. 6. Visualization of the features using t-SNE. (a) MTFCC + transformation. (b) Only data transformation. (c) Using raw data.

4) Visualization: The high-dimensional representations of test loops under three strategies: 1) MTFCC + transformation; 2) only transformation; and 3) raw data; are mapped to a 2-D space using the t-distributed stochastic neighbor embedding (t-SNE) technique. Fig. 6 shows the feature distribution. Class 1 is related to nonstiction loops, and Class 2 is related to stiction loops. It can be seen that in the Fig. 6(a), the samples of Class 1 are mostly located on the right and upper side and the samples of Class 2 are distributed on the lower left side, which means the most discriminate representations are obtained using our proposed MTFCC learning and time-series transformation methods, and even the linear classifier can achieve high accuracy. Moreover, although the samples in the Fig. 6(b) and (c) are difficult to be identified using a linear classifier, the representations shown in Fig. 6(b) are also better since the samples of Class 2 are not mixed with samples of Class 1. The visualization results also show the effectiveness of our proposed method.

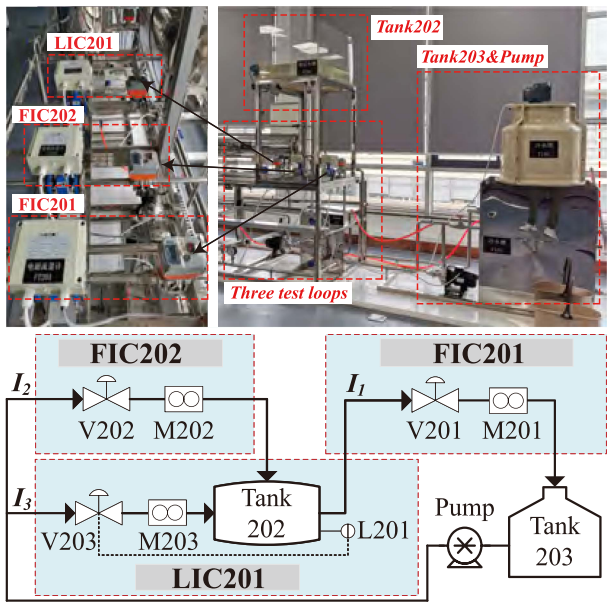


Fig. 7. Real hardware experimental system.

IV. Practical Application

In the practical application, the control loops collected from the hardware experimental system and the real industrial environments are used to verify our detection framework.

The hardware system consists of a liquid level loop and two flow loops. The flowchart and the experimental system are shown in Fig. 7. FIC201 and FIC202 are flow loops, and LIC201 is the level loop. V201, V202, and V203 represent the valves, and M201, M202, and M203 represent the magnetic flow meters. L201 represents a pressure sensor that measures the bottom pressure of Tank 202, and then the pressure value is transformed to the liquid level. V203 and V202 control the water flow into the Tank202, and V201 controls the water flow out of the Tank202. The four other control loops, PIC23002, FIC3107, FIC2228, and F6304, are collected from real industrial environments, in which PIC23002 is a pressure control loop, and it is affected by unknown external disturbances, FIC3107 is a flow control loop and its state is normal, FIC2228 and F6304 are flow control loops, and they were recorded as stiction. The raw time-series recordings of these loops are shown in Fig. 8, in which the X-axis represents time (in seconds), and the Y-axis represents the relative value after normalizing the original data to the range [0,1]. For flow control loops FIC201, FIC202, FIC3017, FIC2228, and F6304, the unit is m^3/h. For loop LIC201 and PIC23002, the units are cm and kPa, respectively. Moreover, the temporal distance matrices are provided for each loop under two random time scales.

The primary hyperparameters of the proposed MTFCC include the size of the temporal distance matrix and the number of time scales. The matrix size is set to 28, the number of time scales is set to an integer in the range of [2,10], and the final detection result is the optimal result under the different number of time scales. In a practical application experiment, the ISDB

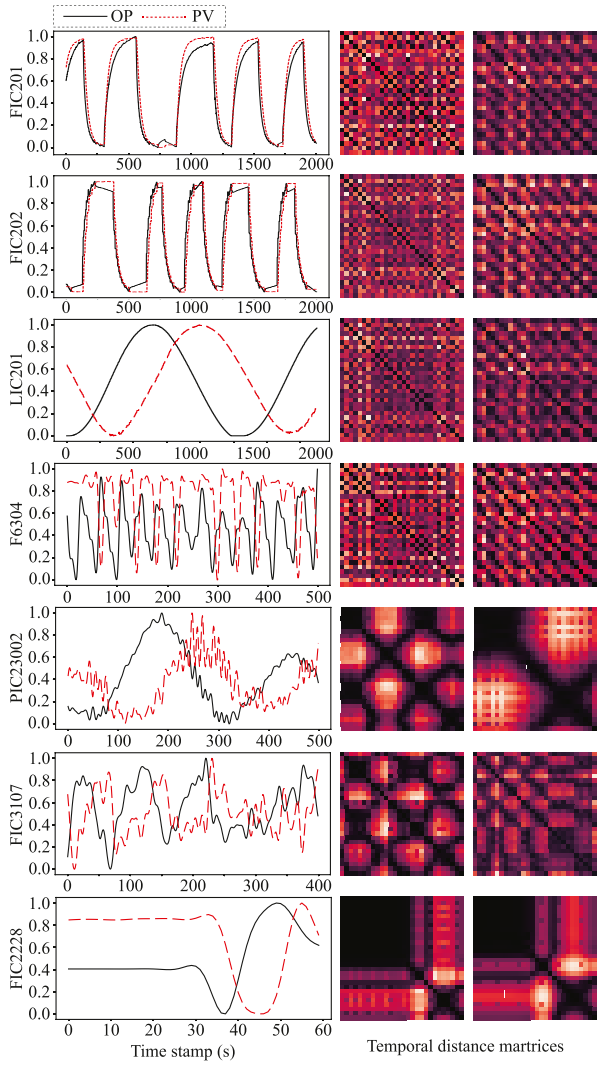


Fig. 8. Raw recordings and visualization of temporal distance matrices for seven real control loops.

benchmark dataset is used as training data, and the test data consists of seven real loops.

In the experiments, we compare the proposed MTFCC method with other seven methods, including LR, random forest (RF), support vector machine (SVM), extreme gradient boosting (XgBoost), LeNet-5, supervised convolutional network (supervised-Conv) [15], and modified stiction detection network (mSDN) [14]. LR and SVM are well-known machine learning algorithms that are simple, effective, and interpretable. Both have a solid theoretical foundation, resulting in wide applications in various fields. RF and XgBoost are ensemble learning algorithms, commonly used techniques in a data science competition since model performance could always benefit from various algorithms. Supervised-Conv, LeNet-5, and mSDN are three DL-based on neural networks. Supervised-Conv is a fully supervised multiscale CNN, which was proposed for valve stiction. LeNet-5 consists of two sets of convolutional and average pooling layers, followed by a flattening convolutional layer, two fully connected layers, and a softmax classifier. mSDN

TABLE V
EXPERIMENTAL RESULTS ON REAL HARDWARE SYSTEM AND INDUSTRIAL ENVIRONMENTS

Loop	Malfunction	MTFCC-based	Supervised-Conv	Lenet-5	mSDN	Xgboost	RF	SVM	LR
FIC201	Normal	Nonstic	Nonstic	Nonstic	Nonstic	Stic	Stic	Stic	Stic
FIC202	Normal	Nonstic	Nonstic	Stic	Stic	Nonstic	Nonstic	Nonstic	Nonstic
LIC201	Normal	Stic	Nonstic	Nonstic	Nonstic	Nonstic	Nonstic	Nonstic	Nonstic
PIC23002	External disturbance	Nonstic	Nonstic	Stic	Nonstic	Nonstic	Nonstic	Nonstic	Nonstic
FIC3107	Normal	Nonstic	Stic	Stic	Stic	Nonstic	Nonstic	Nonstic	Nonstic
FIC2228	Stiction	Stic	Stic	Stic	Stic	Nonstic	Nonstic	Stic	Nonstic
F6304	Stiction	Stic	Stic	Stic	Stic	Stic	Stic	Nonstic	Nonstic

TABLE VI
DETECTION RESULTS ON REAL HARDWARE SYSTEM AND INDUSTRIAL ENVIRONMENTS WITH DIFFERENT NUMBER OF TIME SERIES

Loop	Number of time scales									True label
	2	3	4	5	6	7	8	9	10	
FIC201	0	0	0	0	0	0	1	1	1	0
FIC202	0	0	0	0	0	0	0	0	0	0
LIC201	1	1	1	1	1	1	1	1	1	0
PIC23002	0	0	0	0	0	0	0	1	1	0
FIC3107	0	0	0	0	0	0	0	0	0	0
FIC2228	1	1	1	1	1	0	1	0	0	1
F6304	0	1	0	1	1	0	0	0	0	1

For space considerations, 0 represents the nonstiction class, and 1 represents the stiction class.

is a modified SDN that uses the same model structure but uses a different data transformation method.

The comparison results are shown in Table V and the false detections are underlined. The first three rows of Table V are the experimental results of our methods applied to the real hardware experimental system. These three valves in this hardware system are relatively new, so there are no stiction records. The detection result made by our method is no-stiction for Loops FIC201 and FIC202, which are consistent with the real conditions. However, the inconsistent detection result is given for loop LIC201. The detection results of the other four collected industrial loops are shown in the last four rows. Six comparison methods contain false detection(s), but our method gives the correct detection on these four control loops. Although loop PIC23002 was affected by unknown external disturbances, our method also provides a reliable detection. Overall, our proposed method successfully detects 6 out of 7 loops and achieves high accuracy among all comparison methods.

Another experiment is about the number of time scales. For the benchmark dataset ISDB, the proposed MTFCC benefits from the number of time scales, which means selecting more time scales make obtaining higher accuracy easier. In the practical application, we also present the detection results under the different number of time scales in Table VI. It can be seen that the detection results are optimal at time scales 3, 5, and 6. As the number of time scales increases, the detection results may change from correct to incorrect, such as FIC201 and PIC23002. In other words, the number of time scales is essential to our model. We believe that a reasonable number of time scales will achieve optimal results. The too large or too small number will lead to insufficient or redundant information, respectively. However, automatic selection of reasonable timescales is a challenging topic, which we will explore more in the future.

The last decade has witnessed the great success of DL, and this article proposes a feature learning approach for industrial time-series data based on self-supervised contrastive learning. Then we develop a valve stiction detection framework that enables reliable detection when a small number of labeled data are available. The experimental results on the benchmark dataset and the real industrial environments indicate that the proposed framework is comparative in this monitoring task.

V. CONCLUSION

This article developed a data transformation method and an unsupervised feature learning method for industrial time-series. The key idea is to transform the raw time-series into temporal distance matrices under multiple times scales and encode the distance matrices to the embedding representations by a convolution-based encoder. A general fault detection framework was introduced based on the above methods, and it can be applied to the practical applications in which the raw data are time-series data. Moreover, the proposed framework alleviates the problem of requiring a large number of labeled data under traditional supervised learning frameworks. The valve stiction detection task was considered, and the comprehensive experiments on the industrial benchmark dataset, the experimental hardware system, and the real industrial environments showed the effectiveness of our proposed detection framework. Future work will focus on interpretable representation learning methods for industrial time-series data.

REFERENCES

[1] R. Bacci di Capaci and C. Scali, "Review and comparison of techniques of analysis of valve stiction: From modeling to smart diagnosis," *Chem. Eng. Res. Des.*, vol. 130, pp. 230–265, 2018.
[2] M. Jelali and B. Huang, *Detection and Diagnosis of Stiction in Control Loops: State of the Art and Advanced Methods*. London, U.K.: Springer-Verlag, 2010.
[3] H. Ding, R. X. Gao, A. J. Isaksson, R. G. Landers, T. Parisini, and Y. Yuan, "State of AI-based monitoring in smart manufacturing and introduction to focused section," *IEEE/ASME Trans. Mechatronics*, vol. 25, no. 5, pp. 2143–2154, Oct. 2020.
[4] A. Horch, "A simple method for detection of stiction in control valves," *Control Eng. Pract.*, vol. 7, no. 10, pp. 1221–1231, 1999.

[5] M. Kano, H. Maruta, H. Kugemoto, and K. Shimizu, "Practical model and detection algorithm for valve stiction," *IFAC Proc. Vol.*, vol. 37, no. 9, pp. 859–864, 2004.
[6] Y. Yamashita, "An automatic method for detection of valve stiction in process control loops," *Control Eng. Pract.*, vol. 14, no. 5, pp. 503–510, 2006.
[7] M. Shoukat Choudhury, S. Shah, and N. Thornhill, "Diagnosis of poor control-loop performance using higher-order statistics," *Automatica*, vol. 40, no. 10, pp. 1719–1728, 2004.
[8] M. Rossi and C. Scali, "A comparison of techniques for automatic detection of stiction: Simulation and application to industrial data," *J. Process Control*, vol. 15, no. 5, pp. 505–514, 2005.
[9] Q. P. He, J. Wang, M. Pottmann, and S. J. Qin, "A curve fitting method for detecting valve stiction in oscillating control loops," *Ind. Eng. Chem. Res.*, vol. 46, no. 13, pp. 4549–4560, 2007.
[10] A. Zakharov, E. Zattoni, L. Xie, O. P. Garcia, and S.-L. Jämsä-Jounela, "An autonomous valve stiction detection system based on data characterization," *Control Eng. Pract.*, vol. 21, no. 11, pp. 1507–1518, 2013.
[11] O. Pozo Garcia, A. Zakharov, and S. Jämsä-Jounela, "Data and reliability characterization strategy for automatic detection of valve stiction in control loops," *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 3, pp. 769–780, May 2017.
[12] A. Singhal and T. I. Salsbury, "A simple method for detecting valve stiction in oscillating control loops," *J. Process Control*, vol. 15, no. 4, pp. 371–382, 2005.
[13] B. Kamaruddin, H. Zabiri, A. Mohd Amiruddin, W. Teh, M. Ramasamy, and S. Jeremiah, "A simple model-free butterfly shape-based detection (BSD) method integrated with deep learning CNN for valve stiction detection and quantification," *J. Process Control*, vol. 87, no. 4, pp. 1–16, 2020.
[14] A. A. A. Mohd Amiruddin, H. Zabiri, S. S. Jeremiah, W. K. Teh, and B. Kamaruddin, "Valve stiction detection through improved pattern recognition using neural networks," *Control Eng. Pract.*, vol. 90, pp. 63–84, 2019.
[15] K. Zhang, Y. Liu, Y. Gu, X. Ruan, and J. Wang, "Multiple-timescale feature learning strategy for valve stiction detection based on convolutional neural network," *IEEE/ASME Trans. Mechatronics*, vol. 27, no. 3, pp. 1478–1488, Jun. 2022.
[16] J. W. Dambros, M. Farenzena, and J. O. Trierweiler, "Oscillation detection and diagnosis in process industries by pattern recognition technique," in *Proc. 12th IFAC Symp. Dyna. Control Process Syst.*, 2019, vol. 52, no. 1, pp. 299–304.
[17] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, vol. 119, pp. 1597–1607.
[18] L. Wen, X. Li, L. Gao, and Y. Zhang, "A new convolutional neural network-based data-driven fault diagnosis method," *IEEE Trans. Ind. Electron.*, vol. 65, no. 7, pp. 5990–5998, Jul. 2018.
[19] M. Xia, T. Li, L. Xu, L. Liu, and C. W. de Silva, "Fault diagnosis for rotating machinery using multiple sensors and convolutional neural networks," *IEEE/ASME Trans. Mechatronics*, vol. 23, no. 1, pp. 101–110, Feb. 2018.
[20] Z.-X. Hu, Y. Wang, M.-F. Ge, and J. Liu, "Data-driven fault diagnosis method based on compressed sensing and improved multiscale network," *IEEE Trans. Ind. Electron.*, vol. 67, no. 4, pp. 3216–3225, Apr. 2020.
[21] A. Abanda, U. Mori, and J. A. Lozano, "A review on distance based time series classification," *Data Min. Knowl. Discov.*, vol. 33, no. 2, pp. 378–412, 2019.
[22] J. Lines and A. Bagnall, "Time series classification with ensembles of elastic distance measures," *Data Min. Knowl. Discov.*, vol. 29, no. 3, pp. 565–592, 2015.
[23] H. Yéche, G. Dresdner, F. Locatello, M. Hüser, and G. Rüatsch, "Neighborhood contrastive learning applied to online patient monitoring," in *Proc. 38th Int. Conf. Mach. Learn.*, 2021, vol. 139, pp. 11964–11974.
[24] M. S. Sarfraz, N. Murray, V. Sharma, A. Diba, L. Van Gool, and R. Stiefelhagen, "Temporally-weighted hierarchical clustering for unsupervised action segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 11 220–11 229.
[25] Z. Yue et al., "Ts2vec: Towards universal representation of time series," *Proc. AAAI Conf. Artif. Intell.*, 2022, vol. 36, no. 8, pp. 8980–8987.
[26] W. Qiu, Q. Tang, J. Liu, and W. Yao, "An automatic identification framework for complex power quality disturbances based on multifusion convolutional neural network," *IEEE Trans. Ind. Inform.*, vol. 16, no. 5, pp. 3233–3241, May 2020.
[27] J. Yu and X. Zhou, "One-dimensional residual convolutional autoencoder based feature learning for gearbox fault diagnosis," *IEEE Trans. Ind. Inform.*, vol. 16, no. 10, pp. 6347–6358, Oct. 2020.
[28] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, 2011, vol. 15, pp. 315–323.
[29] M. Shoukat Choudhury, N. Thornhill, and S. Shah, "Modelling valve stiction," *Control Eng. Pract.*, vol. 13, no. 5, pp. 641–658, 2005.
[30] A. S. Brásio, A. Romanenko, and N. C. Fernandes, "Detection of stiction in level control loops," in *Proc. 9th IFAC Symp. Adv. Control Chem. Processes ADCHEM IFAC-PapersOnLine*, 2015, vol. 48, no. 8, pp. 421–426.
[31] Y. Yamashita, "Diagnosis quantification of control valves," in *Proc. SICE Annu. Conf.*, 2008, pp. 2108–2111.
[32] M. Daneshwar and N. Mohd Noh, "Detection of stiction in flow control loops based on fuzzy clustering," *Control Eng. Pract.*, vol. 39, pp. 23–34, 2015.
[33] S. Karra and M. N. Karim, "Comprehensive methodology for detection and diagnosis of oscillatory control loops," *Control Eng. Pract.*, vol. 17, no. 8, pp. 939–956, 2009.
[34] J. W. V. Dambros, M. Farenzena, and J. O. Trierweiler, "Data-based method to diagnose valve stiction with variable reference signal," *Ind. Eng. Chem. Res.*, vol. 55, no. 39, pp. 10316–10327, 2016.

Kexin Zhang received the B.S. and M.S. degrees in engineering from the Chine University of Geosciences, Wuhan, China, in 2016 and 2019, respectively. He is currently working toward the Ph.D. degree in engineering with the College of Control Science and Engineering, Zhejiang University, Hangzhou, China.

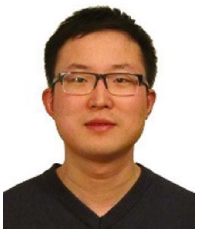
His major research interests include data mining and machine learning on industrial time-series data.

Yong Liu received the B.S. degree in computer science and engineering and the Ph.D. degree in computer science from Zhejiang University, Hangzhou, China, in 2001 and 2007, respectively.

He is currently a Professor with the Institute of Cyber Systems and Control, Department of Control Science and Engineering, Zhejiang University. He has authored or coauthored more than 30 research papers in machine learning, computer vision, information fusion, and robotics. His current research interests include machine learning, robotics vision, information processing, and granular computing.

Yong Gu received the B.S. and Ph.D. degrees in control theory and engineering from Zhejiang University, Hangzhou, China, in 1996 and 1999, respectively.

He is currently an Associate Professor with the Institute of Cyber Systems and Control, Department of Control Science and Engineering, Zhejiang University. He has authored or coauthored more than ten research papers in advanced process control, system identification, soft sensor, and artificial neural networks. His current research interests include machine learning, information processing, advanced process control, and optimization.



Jiadong Wang received the B.S. and M.S. degrees in control engineering from the East China University of Science and Technology, Shanghai, China, and the Ph.D. degree in electrical and computer engineering from the University of Alberta, Edmonton, Canada, in 2005, 2008, and 2013, respectively.

He is currently a Product Manager with SUPCON, Hangzhou, China, and he is also an Adjunt Professor with the Nanjing Tech University, Nanjing, China. His major research interests include PID and MPC control theory and applications.

Xiaojun Ruan received the B.S. degree in chemical engineering from the Beijing University of Chemical Technology, Beijing, China, 2015 and the M.S. degree in chemical engineering with IT from Loughborough University, Leicestershire, U.K., 2016.

He is an Algorithm Engineer with Zhejiang Supcon Software Co., Ltd., Hangzhou, China. His major research interests include data mining, process control, and optimization.